1

AFIT/GAE/ENY/91D-10

IMPROVEMENT OF THE THERMODYNAMIC MODEL FOR A FLUX-DIFFERENCE-SPLITTING ALGORITHM FOR THE COMPUTATION OF HIGH SPEED FLOWS

THESIS

Mark E. Schieve, Captain, USAF

AFIT/GAE/ENY/91D-10

Acknowledgement

My deepest gratitude goes out to my wife, Christine, for her super support and understanding. With her full load of classes and a house move a week before this thesis was due, she definitely outdid herself. I sincerely thank Captain Doty for his patience during my many unannounced visits and for keeping me on track during my research. His enthusiasm and encouragement motivated me throughout my studies.

Table of Contents

List of Figures

List of Symbols

γ	specific heat ratio
ε	shock wave angle
$\Delta\zeta$	axial step size in transformed coordinate system
ζ	transformed axial coordinate
η	transformed radial coordinate
η_x	partial derivative of transformed radial coordinate with respect to x
η_y	partial derivative of transformed radial coordinate with respect to y
θ	flow angle
ν	Prandtl-Meyer angle
ρ	density
ρe	total internal energy
σ_i	flow slope in region i
ϕ_i	entropy parameter in region i
ϕ_o	entropy parameter at some reference temperature
ψ	general variable for Riemann problem
a	speed of sound
C_i	mass fraction of ith species
c_p	specific heat at constant pressure
c_v	specific heat at constant volume
dE	difference in E flux vector

dF	difference in F flux vector
E	E flux vector
F	F flux vector
h_i	static enthalpy in region i
h_t	stagnation enthalpy
M_i	Mach number in region i
MW	molecular weight
P_i	pressure in region i
R_{gas}	gas constant for a mixture gases
R_{univ}	universal gas constant
s_i	entropy in region i
T_i	temperature in region i
$\hat{u}$	specific internal energy
u	axial velocity component
v	radial velocity component
V	velocity magnitude
x	axial direction
y	radial direction
z	coefficient for linearized approximate Riemann solver

AFIT/GAE/ENY/91D-10

Abstract

This study modifies the thermodynamic model of a previously existing first-order accurate Flux-Difference Splitting (FDS) algorithm for planar, supersonic nozzles. The thermodynamic model is changed from a calorically and thermally perfect gas to a thermally perfect (imperfect) gas, where the flow field is "frozen" or non-reacting. The frozen flow and imperfect gas assumptions more nearly approximate the real behavior of a fluid in supersonic propulsive nozzles. The modified code can now account for specific heats that vary as a function of temperature. Using curve fittings of JANAF thermochemical data, the code can handle nine gas species, as well, to model combustion products entering the nozzle inlet. The marching scheme is not altered in order to retain the robustness and efficiency of the first-order method.

An oblique shock reflection study is done to validate the improved gas model. A low pressure, low temperature case and a high pressure, high temperature case are run. For the first case, the perfect and imperfect models are nearly identical. For the more extreme case, pressure for the perfect gas is 9.4% greater than the exact solution, at the upper boundary, across the shock. An interior flow → [illegible]

nozzle is run for the two cases, with air as the working fluid. Again, the two models give identical results for the low pressure case. For the high pressure case, integrated nozzle thrust for the imperfect gas model is 16% higher than that for the original perfect gas model.

IMPROVEMENT OF THE THERMODYNAMIC MODEL FOR A FLUX-DIFFERENCE-SPLITTING ALGORITHM FOR THE COMPUTATION OF HIGH-SPEED FLOWS

I. Introduction

1.1 Historical Development

From the earliest days of space exploration, scientists and engineers have explored the concept of a single-stage-to-orbit (SSTO) flight vehicle. Such a design could takeoff conventionally from existing runways, enter a low-earth orbit, reenter the earth's atmosphere, and, finally, land conventionally on a runway. To do so with an autonomous vehicle stretches the current limits of technology in the areas of structures, materials, aerodynamics, and propulsion. In particular, the air-breathing, hydrogen-fueled propulsion system currently proposed must perform over the wide range of flight conditions that will be encountered during a typical mission profile. This places special demands on researchers' abilities to design and test propulsion systems before actual flight occurs.

The tools of computational fluid dynamics (CFD) will provide the lion's share of this propulsion testing capability. More specifically, CFD will provide the most economical and, at times, the only means of testing nozzle designs in the high Mach number flight regimes (2:1).

In response to this need for computational design tools to model engine flow fields, a computer code was developed by Doty (3) to model the flow of a calorically and thermally perfect gas as it applies to a two-dimensional, maximum thrust, planar nozzle. Figure 1 shows the location and type of geometry for a superson'c, planar nozzle on a NASP-type vehicle. What remains to be determined, though, is the effect of modeling the nozzle flow with this simplified perfect gas model. At the high temperatures involved (approximately T= 4000 K), specific heats of gas are not constant, as a calorically and thermally perfect gas implies (from here on referred to as a perfect gas), but vary significantly as a function of temperature (1:391). This variation can have a measurable effect on flow field properties and thrust calculations.

1.2 Problem Statement

This research effort incorporates an imperfect gas model (thermally perfect) into a previously existing algorithm, originally based on a perfect gas, to more realistically predict the performance of the installed nozzle.

1.3 Objectives

The following objectives of this study are outlined below:

i. Become familiar with the numerical schemes and solution procedures used in the reference (3) nozzle flow program.

ii. The FDS method requires the solution of discontinuities in the flow field, also known as solving the Riemann problem. With the aid of curve-fitted chemical composition data from an existing software package, implement varying specific heats of gas for hydrogen/air combustion products into the solution of the Riemann problem. The basic numerical solution procedure will remain unchanged.

iii. Validate the modified program against an exact, imperfect, oblique shock reflection test case.

iv. Compare thrust calculations of the perfect gas case to the imperfect gas case. A determination will then be made as to whether the improved model provides enough increase in accuracy to justify the added computational time.

1.4 Assumptions

The primary assumption of this study is that the gas obeys the thermally perfect equation of state:

$$P = \rho RT$$

This relation holds for either a perfect or imperfect gas. Thermochemically, the gas flow composition will be fixed at some initial value before solution of the nozzle flow field begins. This would be the case if chemical reactions could be inhibited while still allowing collisions to occur. Flow properties such as pressure, temperature, density, and specific heats may still vary. This inhibited-reaction flow is referred to as "frozen" flow (9:191-192). This assumption is justified due to the extremely low pressures and high speeds that gas molecules encounter at high altitudes. Low pressure at higher altitudes contribute to large molecular mean free paths (distance between collisions), lessening the chances of collisions resulting in reactions. Additionally, molecules traveling at high speeds relative to some nozzle length scale will have extremely small nozzle residence times and will, therefore, have little time to chemically react with each other.

1.5 Scope

This study will deal strictly with modifying the code's current thermodynamic model. The numerical scheme employed will remain intact. While the program has the capability to design maximum thrust nozzles and predict nozzle performance over a wide range of nozzle parameters, a standard nozzle configuration and specified flight condition will be used during thermodynamic model comparisons.

1.6 Background

The recency of work done in reference (3) meant that the chances of finding a sudden rise in literature on the FDS method and it's application of the Riemann problem were not high. A thorough search of the work done regarding CFD algorithms and their attendant thermodynamic models was performed. No new work has been done in the area of internal flow (nozzles) using a first-order FDS method with local solution of the Riemann problem to handle flow discontinuities. By default, the improvement of its thermodynamic model would be an original effort. But many approaches to thermodynamic modeling were surveyed whether or not they involved FDS.

Computational methods usually use the perfect gas model for validation of the scheme. Subsequent efforts to improve the gas model usually go beyond an imperfect gas with frozen flow and straight to one that can handle finite-rate chemistry and vibrational nonequilibrium (8:1). Many can account for dissociating gases. While improved correlation to physics is the goal, there is an accompanying increase in CPU time which can be very expensive if the code must be run on supercomputers.

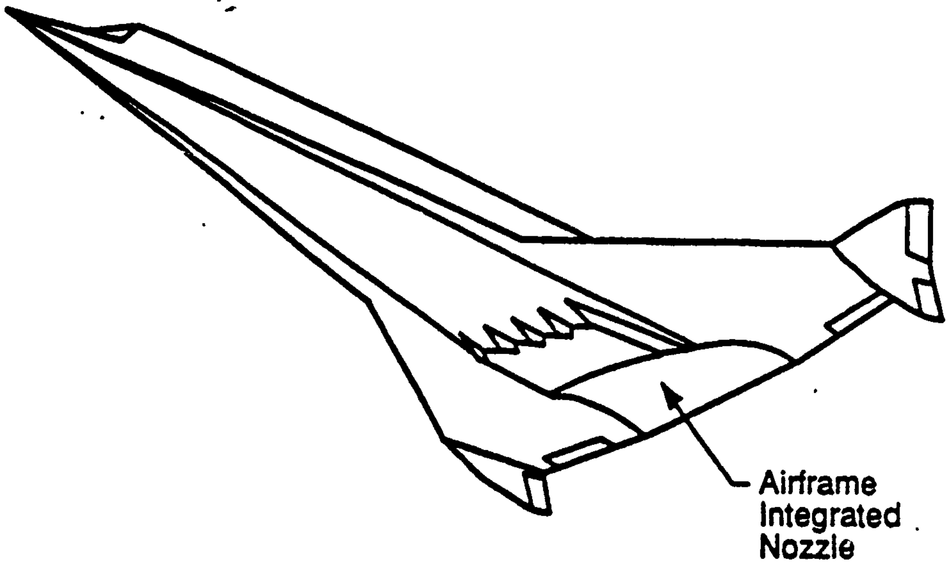

Figure 1 Hypersonic vehicle and planar nozzle (3:5)

II. Governing Equations and Coordinate Transformation

2.1 Governing Equations

The governing equations of motion for a planar, steady, adiabatic, inviscid flow of a compressible fluid with no external work or body forces are the Euler equations, written here in divergence vector form as:

$$\frac{\partial \boldsymbol{E}}{\partial x} + \frac{\partial \boldsymbol{F}}{\partial y} = 0 \qquad (1)$$

where the E and F vectors are given in terms of the conservation variables as:

$$\boldsymbol{E} = \begin{bmatrix} \rho u \\ \rho u^2+P \\ \rho uv \\ u(\rho e+P) \end{bmatrix} \qquad \boldsymbol{F} = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2+P \\ v(\rho e+P) \end{bmatrix} \qquad (2)$$

The first term of each vector represents continuity, the second and third are the x and y components of momentum, respectively, and the fourth is the energy equation. The governing vector equation holds true for both a perfect and imperfect gas.

2.2 Thermodynamic Model

The original code uses a thermodynamic model based on a thermally and calorically perfect gas. As such, the thermal equation of state for this perfect gas assumption is:

$$P = \rho RT \tag{3}$$

The total internal energy for a perfect gas is:

$$\rho e = \frac{P}{\gamma - 1} + \frac{1}{2}\rho(u^2+v^2) \tag{4}$$

The thermally perfect and calorically imperfect gas (imperfect gas) assumptions and equations are presented in Appendix A. The thermal equation of state, eq. (3), remains valid for an imperfect gas. But the total specific internal energy shown uses perfect gas assumptions to obtain the first term on the right hand side. The total internal energy for an imperfect gas is:

$$\rho e = \rho h - P + \frac{1}{2}\rho(u^2+v^2) \tag{5}$$

Static enthalpy, h, is obtained from a least squares curve fit of thermochemical data from JANAF tables (4:32). The enthalpy of the mixture is given as (10:50):

$$h = h_o + \int_{t_o}^{t} c_p dt \tag{6}$$

where the specific heat, c_p, given as a function of temperature and the least squares coefficients, is of the form (4:32):

$$c_p = (a + bT + cT^2 + dT^3 + eT^4)R_{gas} \tag{7}$$

2.3 Coordinate Transformation

The numerical solution is accomplished in the computational plane and requires that the governing equations, eq. (1), be transformed to this space. The physical coordinates x and y are mapped to the computational coordinates as follows (3:142):

$$\zeta = x \qquad \eta = \eta(x,y) \tag{8}$$

The governing equations, eq. (1), are transformed in uniform computational space as (3:8):

$$\frac{\partial(\boldsymbol{E})}{\partial\zeta} = \eta_x \frac{\partial(\boldsymbol{E})}{\partial\eta} - \eta_y \frac{\partial(\boldsymbol{F})}{\partial\eta} \tag{9}$$

For further details on the coordinate transformation, see (3:142).

III. Numerical Algorithm

3.1 Introduction

The Flux-Difference-Splitting (FDS) method is a technique that requires the solution of the Riemann problem to locally model the correct physics of the flow and globally handle complex flow interactions. Figure 2 illustrates the Riemann problem. The Riemann problem is the solution of flow discontinuities, and the waves that result propagate information in specific directions based on the wave angle (3:9). Fluxes across waves can be calculated using known values in region (6) and (0) and the solution aft of the discontinuity. These fluxes are then "split" based on the propagation direction (flow angle), in effect weighting the flow information in the physically correct direction to the next solution plane, i+1. Figure 3 illustrates the computational space.

A first-order accurate marching scheme is used with the FDS algorithm for its monotonic, or nonoscillatory, behavior and robustness. As described by van Leer and reported by Doty, in regions of strong gradients, this is very important (3:10). For further details on the first-order accurate FDS scheme, refer to Appendix C and reference (3). This combination of proper local physics and first-order behavior allows strong gradients (shock waves and contact surfaces) to be handled very accurately (3:2).

3.2 The Riemann Problem

Figure 2 illustrates the setup of the Riemann problem. As described by Doty, Godunov proposed that the general flow property, Ψ, be modeled as a series of uniform flow regions (3:11). The discontinuity is assumed to be located halfway between node points j and j+1 at Riemann node j+1/2. Waves (1) and (3) can be any combination of expansion or compression waves (or neither, in which case there is no turning of the flow). Wave (2) is a contact surface that separates regions (2) and (4). Properties are known at the nodes j and j+1, which correspond to regions (0) and (6), respectively (3:11).

3.3 Solution of the Riemann Problem

The problem consists of solving for the pressure, density, and velocity components in regions (2) and (4). This is done by one of three methods. The first is an exact solution that requires an iterative procedure to solve non-linear, coupled, non-isentropic relations across compression waves (shock waves) and Prandtl-Meyer (simple) expansion waves. Besides the iteration of the non-linear equations, the solution of pressure in regions (2) and (4) must be solved iteratively because contact surfaces can not support normal pressure jumps and the pressures must, therefore, match across the contact surface. The second method is the exact-approximate solution. This is very similar to the exact solution except compression waves are treated as isentropic compressions. Again, non-linear equations, this time due to Prandtl-Meyer relations, must be solved iteratively and contact surface requirements must be met. For details on these exact methods, see reference (3). The last is the linearized-approximate method and is the focus of this study.

3.3.1 Linearized-Approximate Method

This approach eliminates the iteration needed in the previous methods by linearizing the Prandtl-Meyer relations. In addition, compression and expansion waves are assumed to be isentropic. See Appendix B for details on this method. It starts with the differential form of the compatibility relations:

$$\sqrt{M^2-1}\, dP \pm \rho V^2 d\theta = 0 \tag{10}$$

where the velocity magnitude and flow angle are defined as follows:

$$V^2 = u^2 + v^2 \tag{11}$$

$$\theta = \tan^{-1}(v/u) \tag{12}$$

Through manipulation, the linearized equation, for the case where wave (3) is a compression wave and wave (1) is an expansion wave, becomes, respectively:

$$[\ln(P)]_4 + (z_6)\sigma_4 = [\ln(P)]_6 + (z_6)\sigma_6 \tag{13}$$

$$[\ln(P)]_2 - (z_0)\sigma_2 = [lnP)]_0 - (z_0)\sigma_0 \tag{14}$$

where the linearized coefficient is:

$$z = \frac{(\gamma u^2/a^2)}{\sqrt{M^2-1}} \tag{15}$$

Recall that the pressures and flow slope must match across the contact surface:

$$P_4 = P_2 \tag{16}$$

$$\sigma_4 = \sigma_2 \tag{17}$$

Equations (13), (14), (16), and (17) represent four equations and four unknowns and may be solved simultaneously to yield values for P_4 and σ_4.

3.3.1.1 Perfect Gas.

Solving the simultaneous equations for pressure in region (4), P_4, yields:

$$P_4 = \exp([\ln(P)]_6 + z_6(\sigma_4-\sigma_6)) \tag{18}$$

All terms on the right hand side are known. Properties in region (6) are known since these are initial values at plane i. The z_6 coefficient can be "lagged" in the known region (6) or updated in region (4) to provide a more accurate value of the coefficient. Recall that the slope of the flow in region (4), σ_4, is found from the simultaneous solution of eqs. (13), (14), (16), and (17) and is given as:

$$\sigma_4 = \frac{[\ln(P)]_0 - [\ln(P)]_6 + (z_6)\sigma_6 + (z_0)\sigma_0}{(z_6 + z_0)} \tag{19}$$

The details of this derivation appear in Appendix B and reference (3).

With pressure known aft of the wave, isentropic relations such as:

$$\frac{\rho_4}{\rho_6} = \left[\frac{P_4}{P_6}\right]^{1/\gamma} \tag{20}$$

$$a_4 = [\gamma P_4/\rho_4]^{1/2} \tag{21}$$

can be used to obtain density and speed of sound in region (4). For adiabatic flow, conservation of stagnation enthalpy ($h_{t6} = h_{t4}$) across wave 3 is used to determine Mach number and the velocity components in regions (4). But these rely on a constant specific heat ratio, γ, so another approach must be taken for an imperfect gas. See reference (3) for details of the perfect gas Riemann problem solution.

3.3.1.2 Imperfect Gas.

All the above manipulations and equations from eq. (10) to eq. (18) are valid for an imperfect gas as well. Therefore, to obtain the other property variables in region (4), a relationship is needed that can utilize pressure as the only known value in region (4). The following equations show how this requirement can be met when the waves are approximated as isentropic waves. Across the isentropic wave 3, going from region (6) to region (4), the change in entropy across the shock is zero (10:58):

$$s_4 - s_6 = 0 \tag{22}$$

Integrating the differential entropy equation yields:

$$s_4 - s_6 = \phi_4 - \phi_6 - R\ln\left(\frac{P_4}{P_6}\right) \tag{23}$$

Combining theses two relations results in the entropy parameter in region (4), ϕ_4:

$$\phi_4 = \phi_6 + R\ln\left(\frac{P_4}{P_6}\right) \tag{24}$$

The pressure in region (6), P_6, is known and P_4 was calculated from eq. (18). The entropy parameter in region (6), ϕ_6, is calculated using the least squares coefficients and the known temperature in region (6) (4:32):

$$\phi_6 = [\phi_o + a\ln T_6 + bT_6 + \frac{c}{2}T_6^2 + \frac{d}{3}T_6^3 + \frac{e}{4}T_6^4]\, R \tag{25}$$

With a value for ϕ_4 given by eq. (24), the following equation for ϕ_4, as a function of temperature in region (4), is given in least squares coefficient form (4:32):

$$\Phi_4 = [\Phi_o + a\ln T_4 + bT_4 + \frac{c}{2}T_4^2 + \frac{d}{3}T_4^3 + \frac{e}{4}T_4^4]\, R \qquad (26)$$

This equation can be iteratively solved for temperature in region (4), T_4, and the equation of state then determines density.

For adiabatic flow, the imperfect case also relies on conservation of stagnation enthalpy ($h_{t6} = h_{t4}$) to determine velocity components and Mach number, but static enthalpy must be determined as a function of temperature in order to find the x component of velocity, u. Total stagnation enthalpy in region (6) and (4) is:

$$h_{t_4} = h_{t_6} = h_6 + \frac{1}{2}(u^2+v^2)_6 \qquad (27)$$

The local slope of the flow in region (4), σ_4, is defined as:

$$\sigma_4 = v_4/u_4 \qquad (28)$$

Substituting this equation into eq. (27) and solving for the x velocity component, u_4, yields:

$$u_4 = \sqrt{\frac{2(h_{t_4}-h_4)}{1+\sigma_4^2}} \qquad (29)$$

where static enthalpy in region (4), h_4, can be found as a function of T_4 and the least squares coefficients. More complete details of the imperfect gas Riemann problem solution are contained in Appendix B.

3.4 First-Order Accurate Interior Point

The initial values are known at plane i and the solution is desired at the next plane, i+1. Figure 3 shows the stencil in the solution plane. The first-order accurate FDS method uses positively biased information from Riemann node j-1/2 and negatively biased information from Riemann node j+1/2 to integrate the solution from node (i,j) to (i,j+1). To march the solution in the x direction, a combination of finite difference and flux difference approximations are used for the partial derivatives of the **E** and **F** vectors in the governing equation. This is accomplished by computing the Riemann fluxes and then differencing these fluxes. These flux differences are then propagated or "split" in a particular direction based on the local slope of the flow. It should be noted that, after solving the Riemann problem by the new method described in the previous section, this FDS method remains unchanged. Details of the solution procedure are in Appendix C.

The first-order accurate FDS approximation to the transformed governing equation is (3:15):

$$\boldsymbol{E}_j^{i+1} = \boldsymbol{E}_j^i - \Delta\zeta\eta_x\left[d\boldsymbol{E}_{j-1/2}^+ + d\boldsymbol{E}_{j+1/2}^-\right] - \Delta\zeta\eta_y\left[d\boldsymbol{F}_{j-1/2}^+ + d\boldsymbol{F}_{j+1/2}^-\right] \qquad (30)$$

The first-order accurate FDS boundary point calculations are presented in reference (3).

3.5 Decoding the Solution

With the solution vector integrated from plane i to i+1, the **E** and **F** flux vectcrs, in conservative form, are decoded or reduced to the primitive variable form necessary for the solution of the Riemann problem at the next plane. Details are presented in Appendix D. The **E** vector, in conservative form, was presented in Chapter 2 and is repeated here for convenience:

$$\boldsymbol{E} = \begin{bmatrix} \rho u \\ \rho u^2+P \\ \rho uv \\ u(\rho e+P) \end{bmatrix} \tag{31}$$

This can be written in terms of the **E** vector components, which are known from the marching scheme:

$$E1 = \rho u \tag{32}$$

$$E2 = \rho u^2 + P \tag{33}$$

$$E3 = \rho uv \tag{34}$$

$$E4 = u(\rho e + P) \tag{35}$$

As was mentioned in Chapter 2, the E4 vector differs in total internal energy, ρe, for the two thermodynamic models. For a thermally perfect gas, total internal energy is given as:

$$\rho e = \rho \hat{u} + \frac{1}{2}\rho(u^2+v^2) \tag{36}$$

where, for a perfect gas, specific internal energy, $\hat{u}$, is:

$$\hat{u} = c_v T \tag{37}$$

and for an imperfect gas, specific internal energy is:

$$\hat{u} = h - RT \tag{38}$$

After substituting eq. (37) into eq. (36) and performing several manipulations, decoding of the perfect gas solution provides a quadratic equation for the x component of velocity, u, which is a function of the **E** vector components and γ (3:222):

$$\left[\frac{\gamma+1}{2(\gamma-1)}(E1)\right]u^2 - \left[\frac{\gamma}{\gamma-1}(E2)\right]u + \left[(E4) - \frac{1}{2}\frac{(E3)^2}{(E1)}\right] = 0 \tag{39}$$

Since this equation relies on a constant γ, another approach is required for an imperfect gas. Details of the imperfect gas decoding procedure are presented in Appendix D. Begin by substituting eq. (38) into eq. (36). Substitute the resulting eq. (36) into the E4 vector, eq. (35). After manipulating and substituting eqs. (32) and (34) into eq. (35), the E4 vector component becomes:

$$E4 = h\,E1 + \frac{1}{2}E1u^2 + \frac{1}{2}\frac{E3^2}{E1} \tag{40}$$

This equation has two unknowns: static enthalpy, h, and the velocity component, u. Static enthalpy is a function of temperature and the least squares coefficients. By substituting the thermal equation of state and the E2 component into the E1 component, the x component of velocity, u, can also be cast as a quadratic in terms of the unknown temperature:

$$(E1)u^2 + (E2)u + (E1)RT = 0 \tag{41}$$

An initial guess for temperature can now be put into eq. (41) and static enthalpy, h. If the E4 component is not satisfied, the process is repeated. An iterative process, such as the secant method, can be used to find temperature. The calculation of the other primitive variables is as follows:

$$u = \frac{E2 \pm \sqrt{(E2)-4(E1)((E1)RT)}}{2(E1)} \tag{42}$$

$$v = \frac{E3}{E1} \tag{43}$$

$$P = E2 - (E1)u \tag{44}$$

$$\rho = \frac{E1}{u} \tag{45}$$

In summary, the Riemann problem is solved at all the half node locations at plane i by one of the three solution methods described. The fluxes are differenced and split and then marched by the first-order accurate method shown. At plane i+1, the solution vector, E, is decoded to start the process again.

This approach differs from the more familiar marching algorithms. Where decoding the E vector into primitive variables is optional for methods such as the Roe scheme, the Riemann problem requires it. Again, the purpose of this is to model, more directly, the correct physics of the local flow.

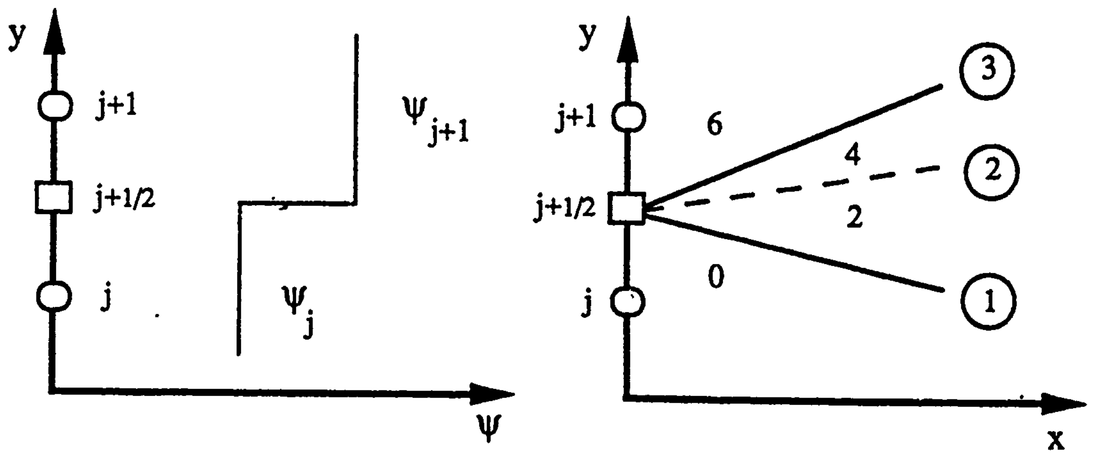

Figure 2 Riemann problem and wave pattern (3:174)

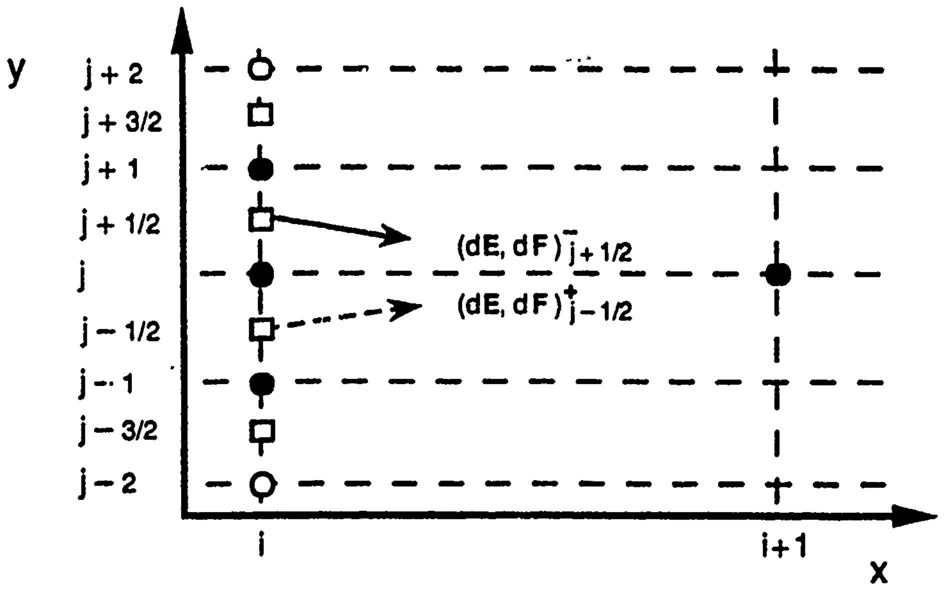

Figure 3 Stencil for first-order accurate FDS method (3:174)

IV. Validation Studies

4.1 Introduction

The imperfect gas model was validated by comparing results to exact solutions. There is no exact solution for the flow field in a hypersonic nozzle. But there are exact solutions available for various flow geometries. One is the oblique shock reflection. Reference (10) has a parallel development structure that discusses perfect gas relations and the corresponding imperfect gas case for normal shocks, oblique shocks, and Prandtl-Meyer waves. The oblique shock case was chosen since the large jump in properties across such a shock are a good test of the numerical method. For details on validation of the perfect gas algorithm, using all three Riemann solvers, for supersonic source flow and Prandtl-Meyer flow, see reference (3).

4.2 Oblique Shock Reflection Study

The geometry for the oblique shock reflection study is shown in Figure 4. The height of the channel is 0.0254 meters before the ramp and the length of the channel is 0.11 meters. Computations begin at point O and proceed to a location past the second dashed line. Uniform flow travels left to right and encounters an abrupt turn at the 10° ramp, depicted as θ_{wall}. The resulting incident shock impinges the ceiling of the channel and a reflected shock is produced. The region used for validation is between the dashed lines. A relatively fine grid of 51 node points, in the y direction, is used for all cases. Grid spacing in the x direction is based on the Courant-Friedrichs-Lewy (CFL) stability criteria (3:144). Details on grid generation can be found in reference (3).

Two initial conditions were run for comparison of the two thermodynamic models. Input for the initial value (IV) line consists of the x and y location of the IV line, static pressure, static temperature, Mach number and flow angle. The data for initial condition #2, IC 2, was obtained from a cycle analysis code that outputs species concentrations and flow properties (6). Table I shows the two initial conditions.

For the validation studies, air was the working fluid in both thermodynamic models. The perfect gas model is very limited in its ability to model different fluids. This is accomplished in the input deck by specifying the specific heat ratio, γ, and the gas constant, R_{gas}, where R_{gas}= R_{univ}/mole-wt. The perfect gas model sets γ= 1.4 and R_{gas}= 287.0, for the validation study. The imperfect gas model uses a more chemically correct air composition in the familiar mass fractional quantities:

$$N_2 = .7556 \frac{kg_{N_2}}{kg_{mix}}$$

$$O_2 = .2316 \frac{kg_{O_2}}{kg_{mix}}$$

$$Ar = .0128 \frac{kg_{Ar}}{kg_{mix}}$$

The gas constant is a function of the molecular weight of the gas and is, therefore, the same value as that for the perfect gas, but the specific heat ratio is a function of temperature. Recall that the fluid composition is unchanged or "frozen" throughout the flow field.

Two areas in the flow were studied. The first was the ceiling of the channel or the top most node. Here, it could be seen whether the full pressure rise on both sides of the shock impingement point would be captured and located correctly. The second was at interior node j= 40 (node counting ascends from the floor to the ceiling). This interior node location will show the capture and location of the incident and reflected shock waves.

The exact solution of an oblique shock wave used is based on an imperfect gas. This solution requires that a double iteration be performed on values of the shock wave angle, ε, and density, ρ. Figure 5 illustrates the oblique shock wave geometry. Details of this method are found in reference 10 (10:369). A short code was developed to quickly find properties aft of an imperfect oblique shock to use as input for the initial value line of the code. This program is located in Appendix E.

Figure 6 shows the results of IC 1 for the ceiling boundary condition. The two models show no discernable difference in pressure between the two solutions. Both models correctly capture the magnitude of the pressure jump and locate the shock well. The agreement is expected given the low pressure and low temperature initial conditions. At temperatures less than T= 600K, γ remains relatively constant and the two models should behave similarly (1:441). This holds true for T= 578 K, where, in region 3, γ= 1.4 for the perfect gas and γ= 1.39 for the imperfect gas. Looking at Figure 7 for the interior node, again, the pressure magnitudes and locations of the shocks are computed correctly.

Figure 8 shows the results for IC 2 at the ceiling. For this boundary, the imperfect gas correctly captures the magnitude and location of the pressure jump, but the perfect gas misses both the location and the pressure jump across the shock by a significant amount: a 9.4% pressure difference. The difference between the thermodynamic models can be attributed to the greater effect that the higher temperature has on γ. For a temperature, T= 4278 K, in region 3, γ= 1.276 for the imperfect gas and γ= 1.4 for the perfect gas. This is due to the perfect gas having a higher percentage of energy conversion, across the shock, show up as translational molecular energy than would occur for an imperfect gas. Since temperature is a direct measurement of the translational energy mode, the perfect gas model has a greater magnitude change in pressure and temperature across the shock. For details on the mechanisms of high-temperature effects for an imperfect gas, see Appendix A. The perfect gas model misses the location of the shock because it computes a shock wave angle, ε, that is greater than that for the imperfect gas. The shock wave angle is, therefore, located a shorter distance in the x direction along the upper wall.

Figure 9 shows the results for the interior node for IC 2. Again, the imperfect model correctly captures the shock location and pressure magnitude rise for both waves. The pressure difference between the exact solution and the perfect gas, in region 2, is 5.3% and in region 3, 9.4%.

These results also show the superb behavior of the first-order FDS method. The most obvious characteristic is the monotonic behavior of the method. No overshoot of the pressure magnitude occurs as does with some second-order accurate methods that have not been modified to detect gradients and reduce oscillations (3:24).

The preceeding validation studies have shown that the improved thermodynamic model has been correctly implemented into the FDS method and provides consistent results. More physically correct solutions are now possible for the conditions likely to be encountered in SCRAMJET nozzles.

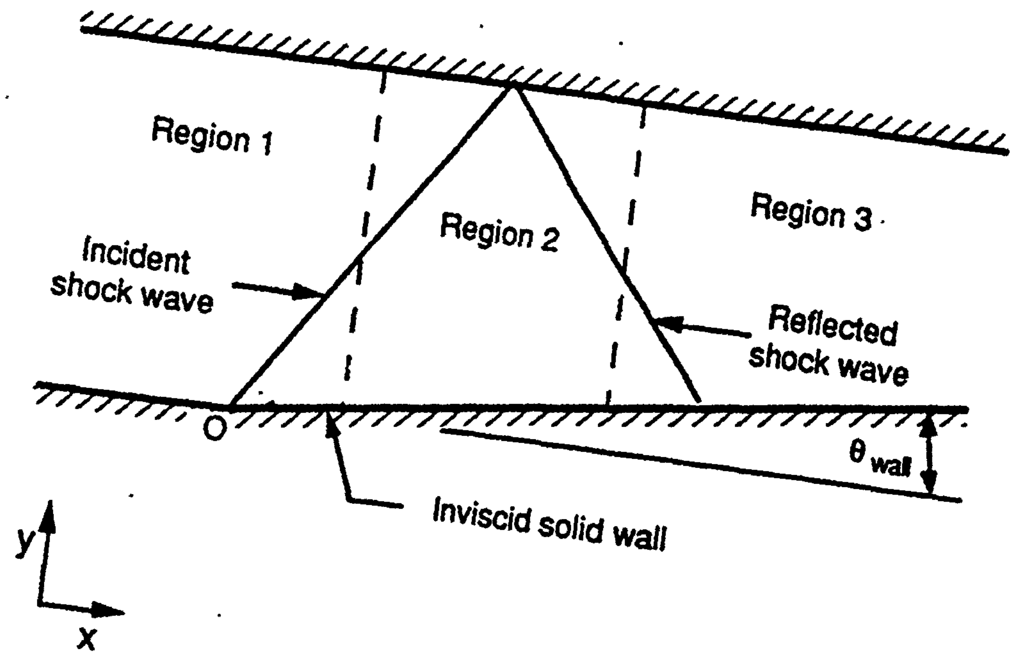

Figure 4 Geometry for shock wave reflection study (3:30)

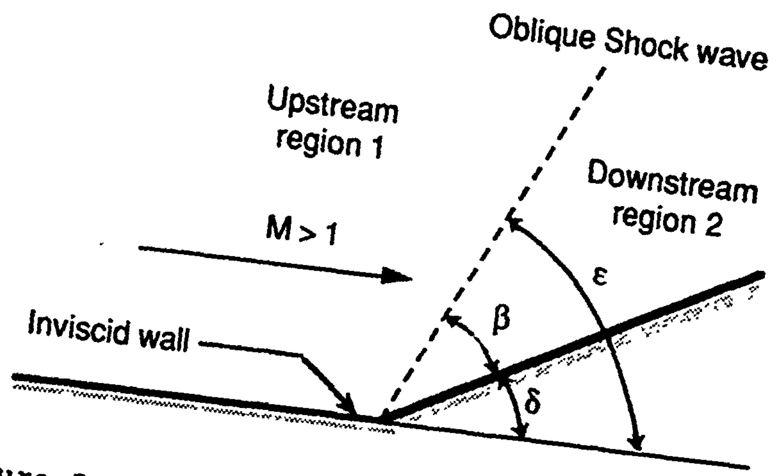

Figure 5 Oblique shock wave geometry (3:174)

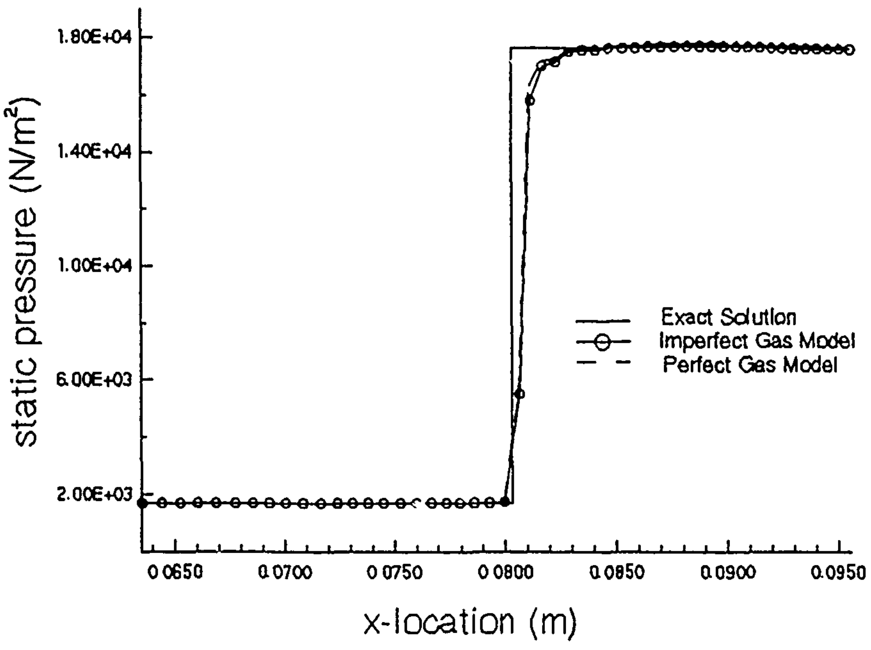

Figure 6 Pressure along upper wall for IC 1

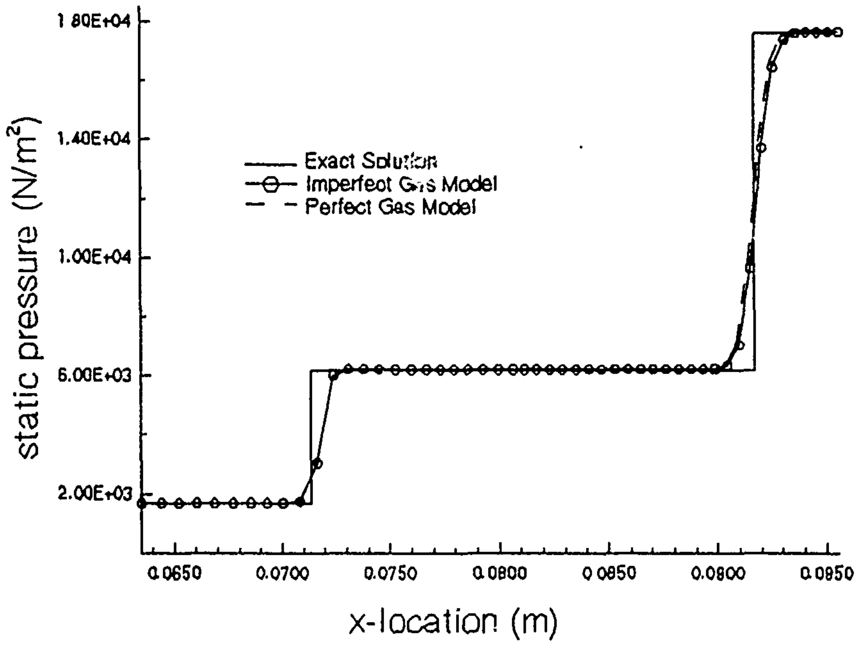

Figure 7 Pressure at interior node j=40 for IC 1

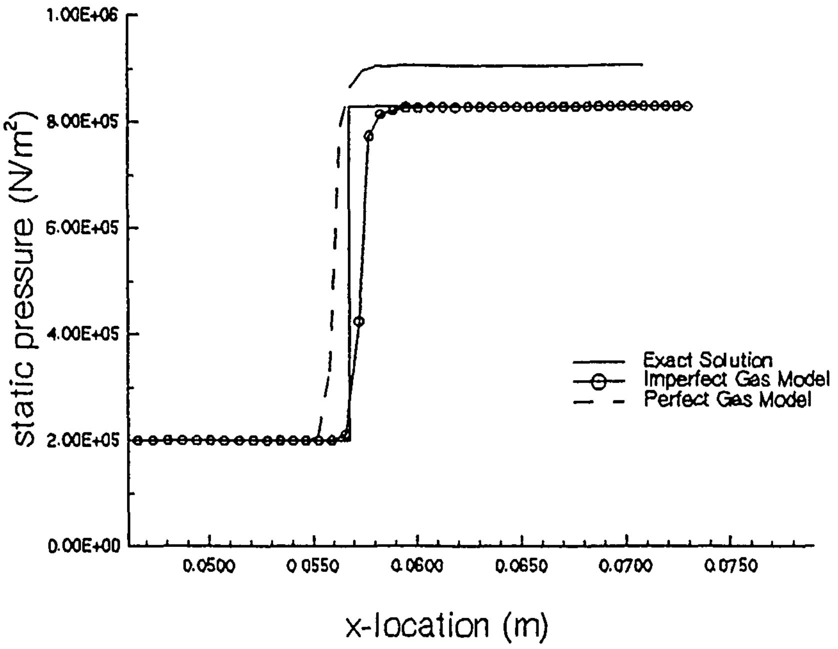

Figure 8 Pressure along upper wall for IC 2

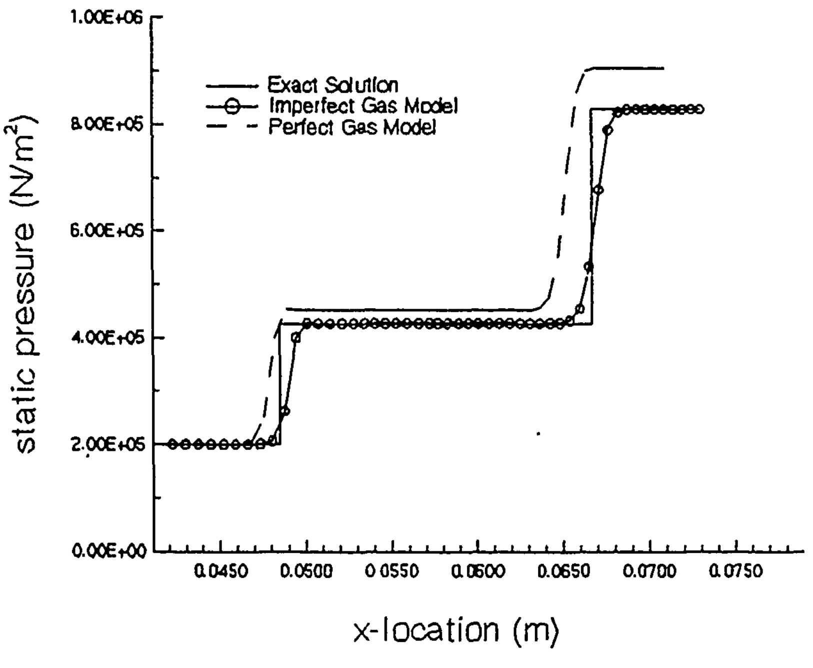

Figure 9 Pressure at interior node j= 40 for IC 2

Table 1. Initial conditions for oblique shock reflection study

	IC 1	IC 2
ramp angle (θ_{wall})	10°	10°
Mach Number	6.0	3.465
pressure (N/m^2)	1696.4	200703
temperature (K)	273.23	3079.1

V. Results and Discussion

5.1 Introduction

Analysis of different flow conditions for a supersonic planar nozzle is presented in this chapter. The first case is for the internal flow of the supersonic nozzle. The second case involves an interior/exterior flow to simulate actual flight conditions. Static pressure, static temperature, and thrust are plotted as a function of the x location along the nozzle wall.

Doty's code (reference 3) can run a variety of nozzle geometries ranging from a straight wall nozzle to a skewed-parabola nozzle wall. The lower cowl can also be angled at various "hinge" points to further aid nozzle optimization. For both flow configurations, a straight cowl and a parabolic nozzle wall are used.

5.2 Internal Nozzle Analysis

Figure 10 illustrates the internal nozzle geometry. For the internal flow nozzle, the cowl is extended the length of the nozzle, in effect, splitting the flow. An attachment angle, θ_B, must be specified for all nozzles; $\theta_B = 25°$ is arbitrarily chosen for this analysis. Two sets of initial conditions were run. The first set represents a less extreme temperature and pressure. The second represents conditions consistent with hypersonic flight at a freestream Mach number of $M_\infty = 15$. The later conditions were obtained from a cycle analysis code (6). The modified imperfect gas code was designed to simulate nine gas species for a given flow field but the three constituents of air (nitrogen, oxygen, and argon), in the correct mass ratio, were used for comparison purposes. Nozzle inlet conditions are listed in Table 2.

The pressure, temperature, and thrust plots for IC 1 are shown in Figures 11, 12, and 13, respectively. For this low-pressure, low-temperature case, the two gas models give nearly identical results. This behavior is expected since the variation of specific heats with temperature is negligible at these conditions. This is consistent with the IC 1 validation study case. IC 1, therefore, proves that the imperfect gas model is consistent with the proven perfect gas model from reference (3). The plots of pressure and temperature illustrate particular characteristics of the nozzle. The large drop-off in temperature and pressure, in Figures 11 and 12, respectively, is a result of the rapidly expanding flow around the nozzle attachment radius (AB in Figure 10). The subsequent small rise in pressure and temperature, is a result of recompression of the flow in the parabolic region of the nozzle (BC in Figure 10). Figure 13 shows the integrated thrust along the nozzle wall, where both models predict the same result.

Pressure, temperature, and thrust are plotted for IC 2 in Figures 14, 15, and 16, respectively. For this case, noticeable differences are seen between the thermodynamic models.

Temperature shows a greater difference in Figure 15. Specific heat ratio, γ, for the perfect gas is held constant at $\gamma = 1.4$ while γ varies with temperature for the imperfect gas. Notice also that temperature drops a larger amount for the perfect gas. Referring to Appendix A, energy conversion, across expansion or compression waves, is distributed over different molecular energy modes, with a perfect gas receiving a greater percentage of translational energy (ie., temperature) than an imperfect gas does. That is why temperature and pressure have a greater decrease in the expansion region of the nozzle for the perfect gas compared to the imperfect gas. Pressure shows the least difference in Figure 14. Recall that the calculation of pressure for the linear-approximate method was nearly the same for both gas models. The only difference was in the calculation of the z coefficient in eq. (15), where γ is a function of temperature. The coupling of temperature and pressure in the thermal equation of state results in a similar trend for pressure, but to a lesser degree. Consequently, pressure differences will not manifest themselves to a large extent.

A smaller magnitude drop in pressure, for the imperfect gas, results the in greater thrust shown in Figure 16. Recall, in the shock validation study, that the perfect gas model overpredicted the pressure magnitude rise for IC 2. This trend continues for the interior nozzle, except pressure is falling instead of rising. The main point is the overprediction of magnitude changes in properties. In this case, integrated thrust along the nozzle wall for the imperfect gas is sixteen percent greater at the nozzle exit. This is a significant difference and illustrates the effect of accounting for property variations as a function of temperature.

5.3 Internal/External Nozzle Analysis

The internal/external flow configuration was run with air for both the interior and exterior regions. Although the multiple species from combustion products of H_2/air can be simulated, the variable specific heat trends for air are similar. Figure 17 illustrates the nozzle flow and geometry. Freestream conditions consisted of an altitude of 129,900 m, pressure of 303.36 N/m^2, temperature of 250 K, and Mach number of 15.0. Initial conditions for the exterior flow were obtained by sending air across a 10° ramp, simulating flight at some angle of attack. This is a simplification of the forebody compression that would actually occur. These initial conditions will make up the properties in region 2 of Figure 17. Perfect and imperfect gas relations were used for computing external initial conditions for the respective thermodynamic models. Table 3 shows the exterior initial conditions. Table 4 shows the nozzle inlet conditions for the two models. This corresponds to combustor exit flow in region 1 obtained from a cycle code using free stream conditions (6). Figures 18, 19, and 20 prove the modified code works for an internal/external flow case. Trends are reversed compared to the interior flow nozzle. Now the imperfect gas has a larger pressure and temperature magnitude change, shown in Figures 18 and 19, respectively. This is due to the values of γ and R_{gas} (γ= 1.25 and R_{gas}= 332.26) used for the perfect gas model. This reversal demonstrates the sensitivity of the flow field solution to changes in the specific heat ratio and the gas composition, which manifests itself in R_{gas}. Though not easily seen in Figure 18, pressure for the perfect gas is slightly higher than that for the imperfect gas. The coupling trend between pressure and temperature, where temperature shows the larger diference between the thermodynamic models (Figure 19), is the same as that seen for the interior nozzle. The modified code can not yet handle one mixture for the interior and another for the exterior flow, so no comparisons are made between the two thermodynamic models or with the interior nozzle.

Central processing unit (CPU) run time was obtained for the internal/external flow geometry to compare computational efficiency. The perfect gas model required 2.1 seconds of CPU run time while the imperfect gas model finished the solution in 7.9 seconds. This difference is a result of the iterative procedures needed to determine properties aft of the waves for solution of the Riemann problem. Considering the already remarkable performance of the first-order accurate FDS scheme, this is not a large penalty to pay for a more physically correct solution.

5.4 Summary

Along with the validation study of oblique shock reflections, the interior flow nozzle, again, demonstrates the effect of accounting for the caloric imperfections of high-temperature gases. Since the net thrust margins required to launch a NASP-type vehicle are very small, the thrust difference, found in the proceeding nozzle study, are significant. Realistic first-order analysis of hypersonic nozzles is important to final nozzle design.

The imperfect gas model does not account for all the physics of the flow. The frozen or non-reacting flow assumption made here does not account for dissociation losses nor energy recovery due to recombination. But a study by Snelling (7), which implemented a finite-rate chemistry model into high-expansion rate flow fields and nozzle geometries similar to ones used here, concluded that there is little difference between the frozen flow assumption and the finite-rate chemistry model in solving the flow field. This supports the fact that the improved thermodynamic model can more realistically model the actual physics of such flow fields. Where differences turn out to be small between thermodynamic models, the first-order accurate, imperfect gas FDS method should be significantly more efficient than a finite-rate algorithm in terms of CPU time.

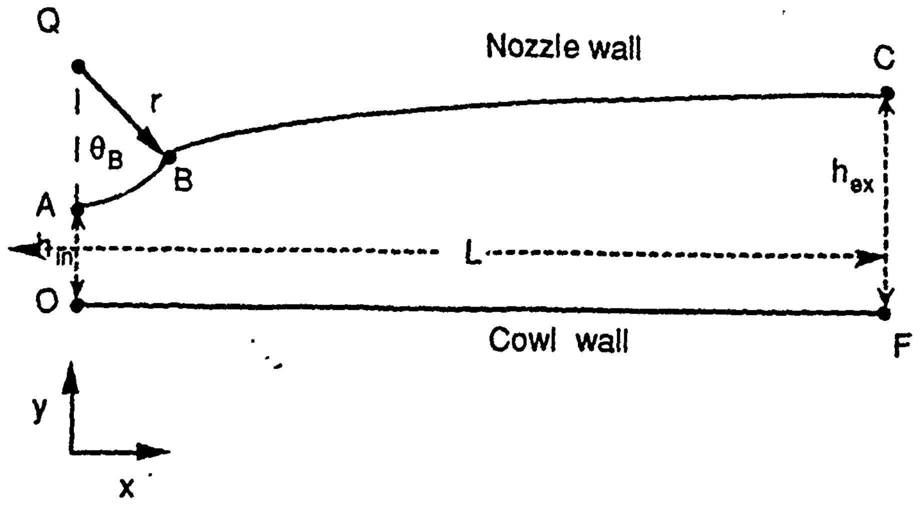

Figure 10 Parabolic nozzle contour (3:46)

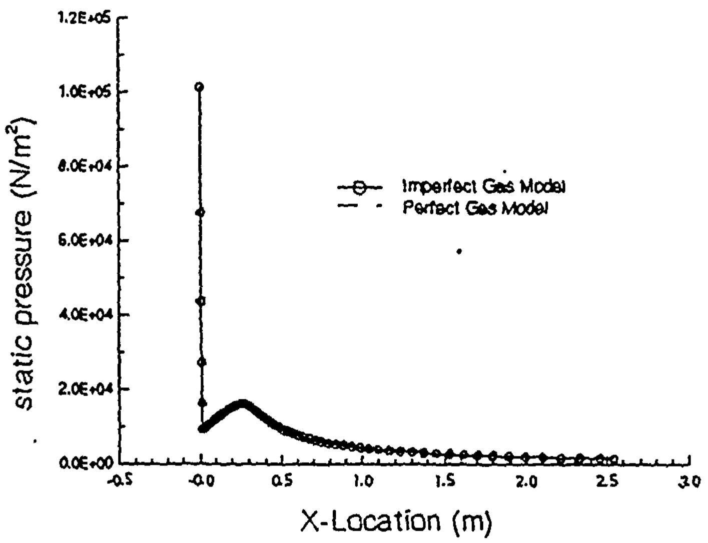

Figure 11 Pressure along nozzle wall, IC 1

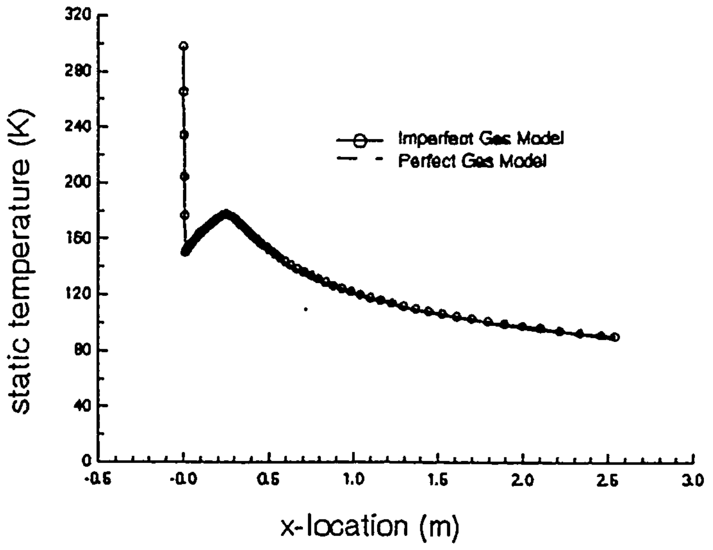

Figure 12 Temperature along nozzle wall, IC 1

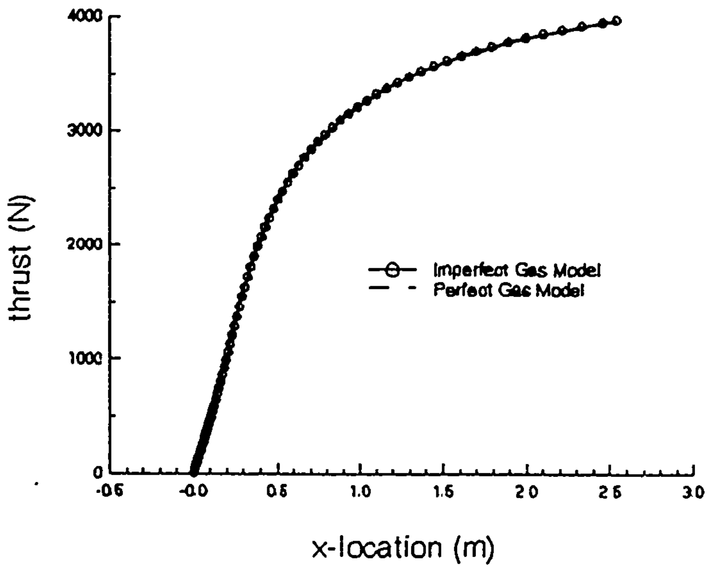

Figure 13 Interior nozzle thrust, IC 1

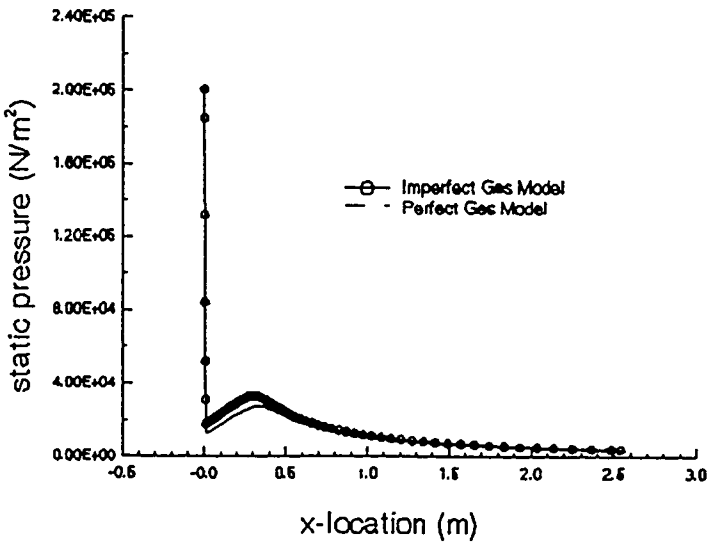

Figure 14 Interior nozzle pressure, IC 2

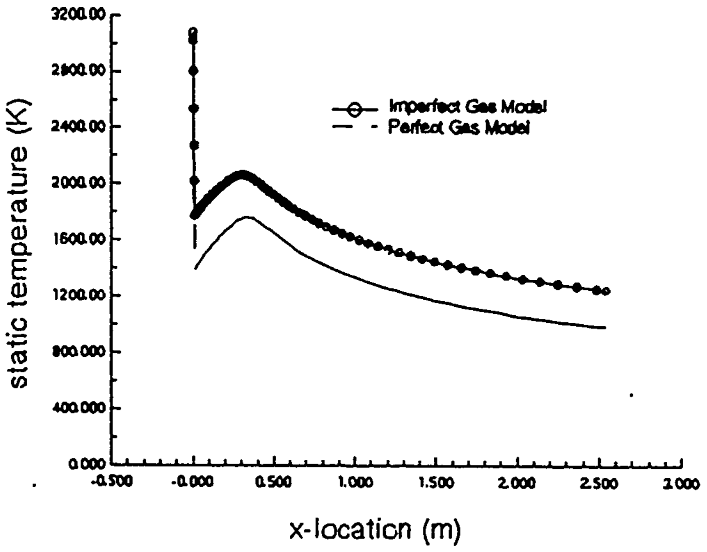

Figure 15 Interior nozzle temperature, IC 2

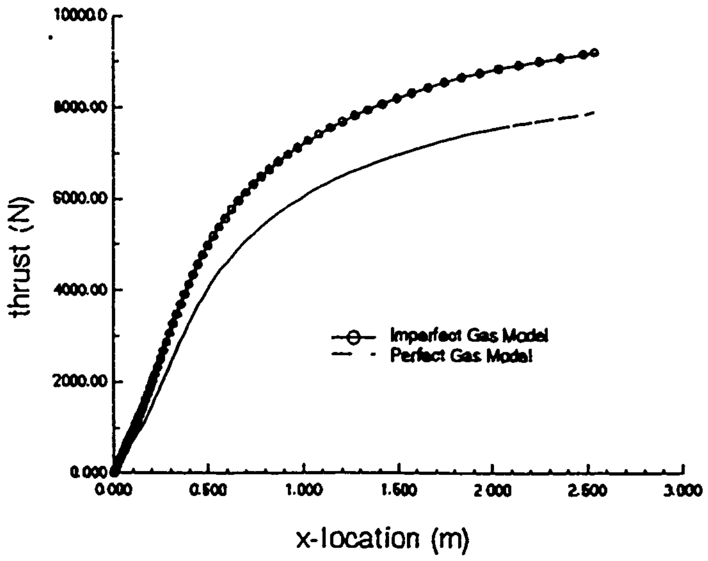

Figure 16 Interior nozzle thrust, IC 2

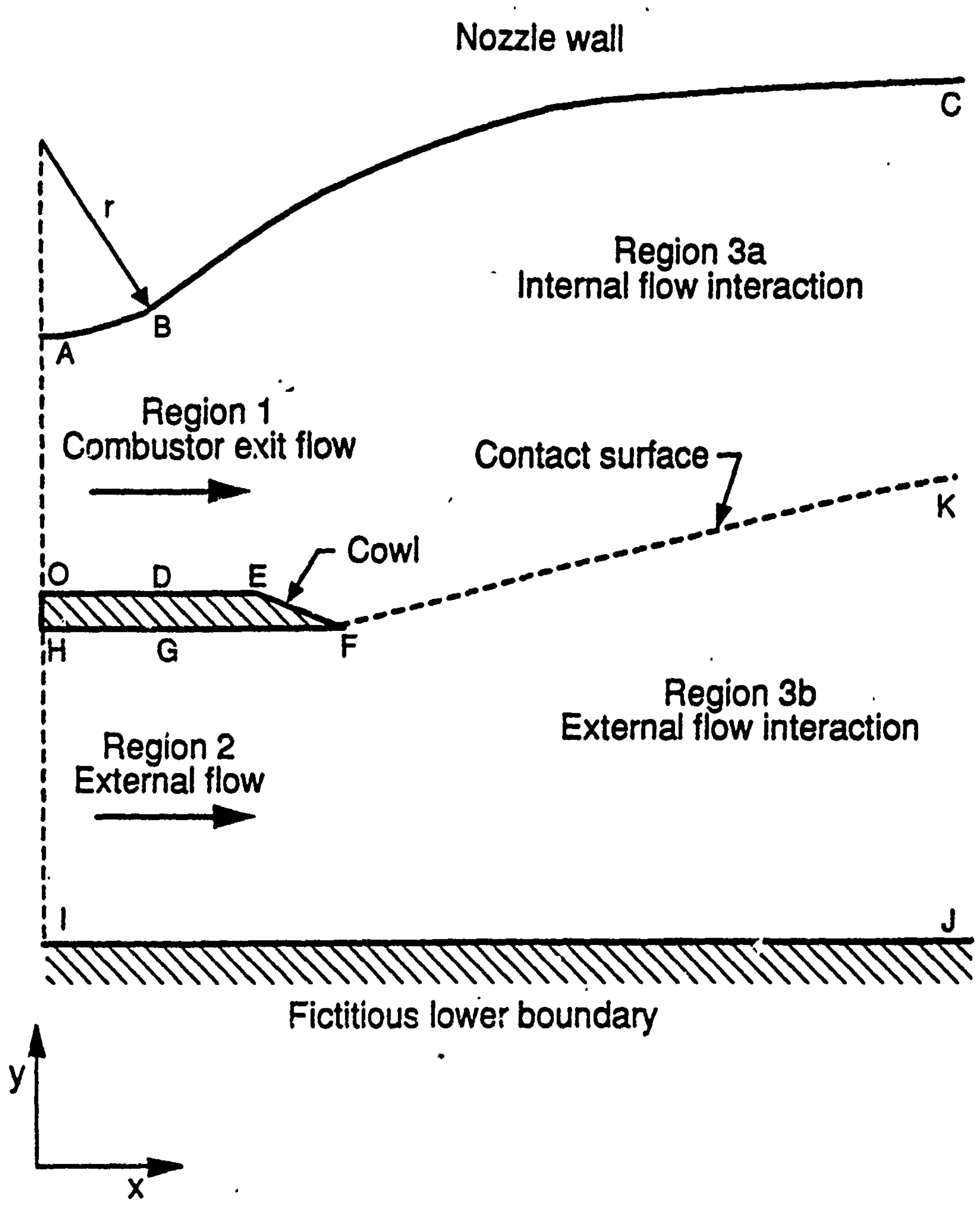

Figure 17 Internal and external nozzle flow and geometry (3:60)

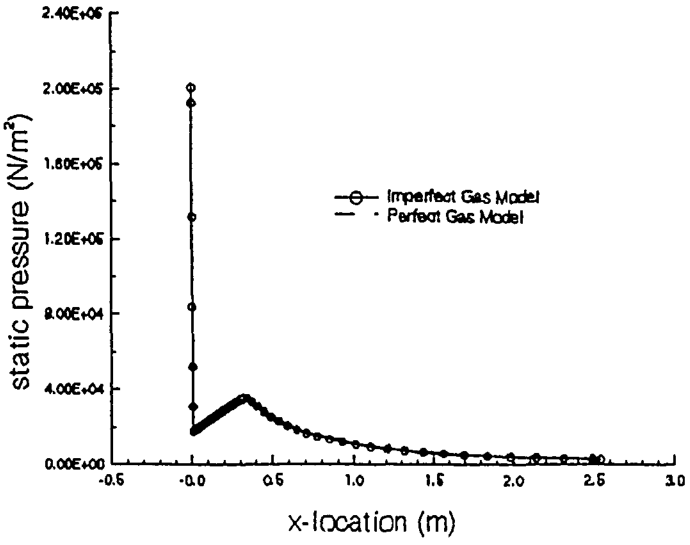

Figure 18 Pressure along nozzle wall for internal/external flow nozzle

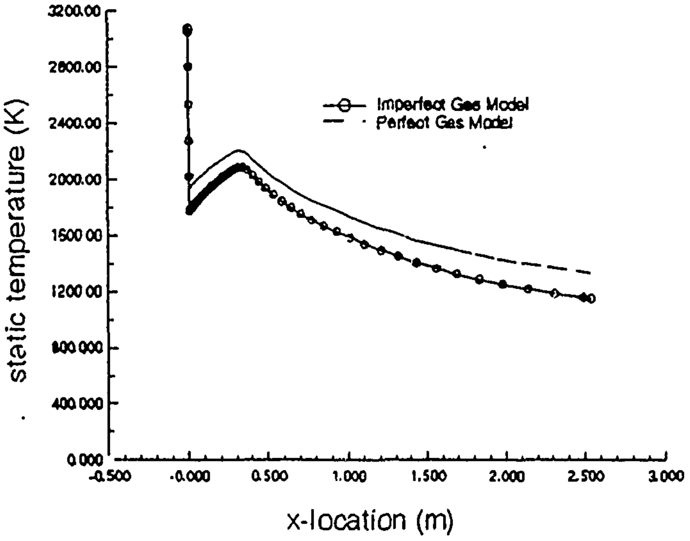

Figure 19 Temperature along nozzle wall for internal/external flow nozzle

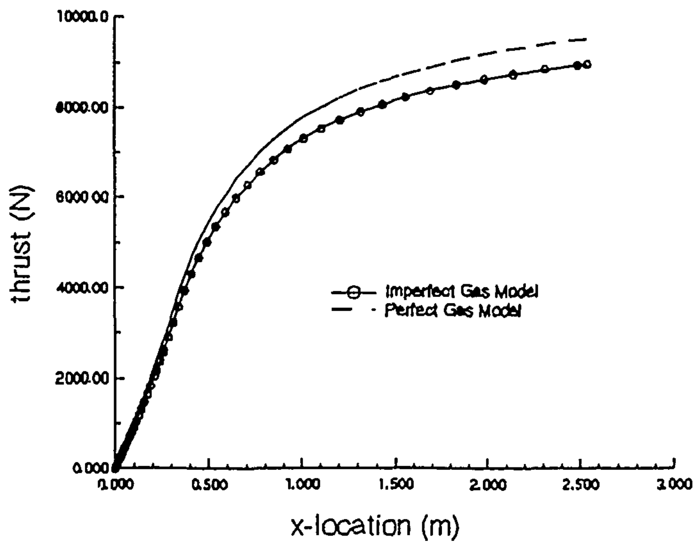

Figure 20 Integrated thrust along nozzle wall for internal/external nozzle

Table 2 Initial conditions for internal flow nozzle

	IC 1	IC 2
flow angle (θ)	0°	0°
Mach number	6.0	3.465
static pressure (N/m^2)	1696.4	200703
static temperature (K)	273.23	3079.1

Table 3 Exterior initial conditions for the internal/external flow nozzle

	Perfect Gas	Imperfect Gas
Mach number	8.16	8.42
static pressure (N/m^2)	4095.6	4068.9
static temperature (K)	800.2	776.4
specific heat ratio, γ	1.4	1.4
gas constant, R_{gas} (J/kg/K)	287	287

Table 4 Interior initial conditions for the internal/external flow nozzle

	Perfect Gas	Imperfect Gas
Mach number	3.465	3.465
static pressure (N/m^2)	200703	200703
static temperature (K)	3079.1	3079.1
specific heat ratio, γ	1.25	1.28
gas constant, R_{gas} (J/kg/K)	332.26	287.0

VI. Summary and Recommendations

6.1 Summary

The thermodynamic model of a previously existing CFD algorithm has been improved by implementation of a thermally perfect and calorically imperfect gas model. By modifying the thermodynamic model and maintaining the desirable characteristics of the original marching scheme, a more physically correct solution of supersonic planar nozzle flow is available.

The imperfect gas model was shown to be nearly identical to exact solutions that were used in the oblique shock reflection validation studies. Physically real flow geometries, as well as the validation studies, revealed that the perfect gas model overpredicted magnitude changes in performance quantities as pressure and temperature were increased. CPU run times were 3.75 times longer for the imperfect gas but the advantages of a more physically correct solution outweigh the reduction in computational efficiency. The full potential of the imperfect gas model will be realized when combustion species can be run for internal nozzle flow and air for external flow.

In the design and optimization of a hypersonic nozzle, the differences between thermodynamic models will show up in the size of the nozzle required to provide a specified thrust. In the design of a NASP-type vehicle, these results will, in turn, influence engine size, fuel onboard, vehicle size, and so on. Therefore, new and more accurate thermodynamic models will be crucial to optimizing the entire vehicle design.

6.2 Recommendations

Inevitably, follow-on research topics arise as a by-product of such a study. Some future areas of interest to accompany this research would be:

1. Further modification of the thermodynamic model to include finite rate chemistry and equilibrium and nonequilibrium flows.
2. Experimental analysis of a simple nozzle geometry to validate the imperfect gas model in the FDS algorithm.
3. An accuracy and efficiency comparison of the modified FDS algorithm with a parabolized Navier-Stokes code.
4. Design, optimization, and comparison of planar supersonic nozzles using the perfect and imperfect thermodynamic models.

Appendix A: Thermodynamic Model

A.1 Introduction

The complexity of a thermodynamic model depends on how much physics of the flow one wishes to account for. A perfect gas is the simplest, where specific heats, c_v and c_p, and hence, the specific heat ratio, γ, remain constant throughout the flow. In this case, enthalpy, h, and specific internal energy, $\hat{u}$, are explicit functions of temperature (1:388):

$$h = c_p T \tag{47}$$

$$\hat{u} = c_v T \tag{48}$$

Again, the thermal equation of state holds and for frozen flow, the specific gas constant, R_{gas}, remains constant since the molecular weight of the gas is fixed:

$$R_{gas} = \frac{R_{universal}}{MW} \tag{49}$$

For the perfect gas assumption, many basic thermodynamic relations can be solved for explicitly and shock wave analysis is comparatively straightforward.

At the other end of the spectrum is a nonequilibrium, chemically reacting gas. This model accounts for most high-temperature gas effects. Flow properties are not only a function of temperature but also pressure and time. Time is an independent variable since reactions have not reached steady state. The effect of pressure is to inhibit reactions at higher pressures and promote them at lower pressures (1:374).

Dissociation is another high-temperature effect. Dissociation, which alters the molecular weight of the gas, occurs at around 2500 K for O_2, and begins at 4000 K for N_2. Van der Waals' effect also comes into play as intermolecular forces become important (1:390).

A.1 The Imperfect Gas Model

When categorized, by assumptions, among the various thermodynamic models, an imperfect gas assumes that intermolecular forces are negligible. With respect to a perfect gas, an imperfect gas is the next step in complexity. The difference between these two models is a function of which mode of internal energy is excited. When dealing with monatomic gases, specific heats are basically independent of temperature since only the translational energy mode is available (neglecting electronic energy). Since most flow field gases are diatomic, the translational, vibrational, and rotational energy modes are available, even at room temperature (5:222). For air as a perfect gas, it is assumed that internal energy is only a function of translational and rotational degrees of freedom (in this discussion of energy modes, e will represent specific internal energy instead of $\hat{u}$):

$$\begin{aligned} e_{perfect} &= e_{tr} + e_{rot} \\ &= \frac{3}{2}RT + RT \\ &= \frac{5}{2}RT \end{aligned} \tag{50}$$

Recalling that $e=c_vT$, the specific heat at constant volume yields $c_v=5/2R$. For a thermally perfect gas:

$$\begin{aligned} c_p &= c_v + R \\ c_p &= \frac{7}{2}R \end{aligned} \tag{51}$$

$$\begin{aligned} \gamma &= \frac{c_p}{c_v} \\ \gamma &= \frac{7}{5} \end{aligned} \tag{52}$$

Where the familiar quantity, $\gamma= 1.4$, is obtained for air as a perfect gas (9:124,133).

When temperature reaches 600 K, vibrational energy is no longer negligible (1:441) and a new internal energy mode must be added. From a statistical mechanics approach, this component is represented, without further proof, as (1:439):

$$e_{vib} = \frac{h\nu/kT}{e^{h\nu/kT}-1}RT \tag{53}$$

where h is Planck's constant, k is Boltzmann's constant, and ν is the fundamental vibrational frequency of the molecule.

The imperfect gas model accounts for this vibrational mode.

In summary, four modes of energy each contribute a portion to the total internal energy (1:440):

$$e_{tot} = [\frac{3}{2}RT]_{trans} + [RT]_{rot} + [\frac{h\nu/kT}{e^{h\nu/kT}-1}RT]_{rot} + e_{el} \qquad (54)$$

Comparing this equation to eq. (50)a, it is obvious that the energy of gaseous molecules is spread over more modes for an imperfect gas (three modes) than for a perfect gas (two modes). For perfect gas flow across a shock, the kinetic energy in front of the shock is converted to translational and rotational energy aft of the shock. For imperfect gas flow, the vibrational energy mode can accommodate some of this energy conversion across the shock, lessening the amount of energy going into the translational and rotational modes, relative to a perfect gas. This effect is important at temperatures above 600 K. Since temperature is a direct measurement of kinetic (translational) energy, a perfect gas tends to overpredict temperature (1:510).

A.2 Implementing Imperfect Gas Assumptions

Determining where modifications to Doty's algorithm (3) are required is a matter of locating what calculations are based on a perfect gas model. The most obvious changes are in the solution of the Riemann problem. These are basically compression or expansion wave problems and applying imperfect gas assumptions is well documented in text materials (10). For the linear-approximate solution that is being used for this study, the method for finding pressure aft of the discontinuity remains the same. What is less straightforward is how to calculate properties, such as entropy and enthalpy, in the aft shock region based on variable specific heats. Since decoding of the E and F vectors is based on constant γ, this must also change.

As is mentioned in the beginning, a desirable capability is to handle a multiple species gas. The current perfect gas code simulates air (physically based on a combination of nitrogen, oxygen, and argon) as the working fluid. The only way to alter the composition of the gas is through the gas constant, R_{gas}, where $R_{gas} = R_{univ}$/mole-wt. It does not have the capability to handle individual gas species. To allow greater flexibility, a multiple species gas would allow this code to handle the output from a separate cycle code analysis program. The combustion products of a SCRAMJET engine will consist of mostly water and nitrogen which are quite different from air as far as molecular weight goes, and hence, will behave differently. Doty's code can handle a nonuniform input from a combustor exit and this capability would be enhanced with true combustor products rather than air.

A.3 Complex Chemical Mixtures

A FORTRAN code, referred to here as CET89, by Gordon and McBride (4), outputs, among other things, the least squares coefficients of specific heats, enthalpy, and an entropy parameter as functions of temperature. Data from JANAF thermochemical data has been curve fitted for approximately 400 species of gas using the method of least squares to the obtain coefficients. There are two sets of coefficients for each species that correspond to two temperature ranges, 300-1000 K and 1000-5000 K. The calculation of the three properties are listed below.

A.3.1 Specific Heat, c_p.

Specific heat for a mixture of gases is represented by the following summation (10:50):

$$c_p = \sum_{i=1}^{n} C_i c_{p_i} \tag{55}$$

where n is a particular gas species and C_i is the mass fraction of the species. In terms of the least squares coefficients c_p is written, on a mass basis, as (4:32):

$$c_{p_i} = (a + bT + cT^2 + dT^3 + eT^4) R_{gas} C_i \tag{56}$$

A.3.2 Enthalpy, h.

Enthalpy for a thermally perfect gas is given by the following expression (10:50):

$$h = h_o + \int_{t_o}^{t} c_p dt \tag{57}$$

Substituting eq. (56) into this equation yields the least squares coefficients form of enthalpy, for the ith species, on a mass basis (4:32):

$$h_i = [(h_o + aT + \frac{b}{2}T^2 + \frac{c}{3}T^3 + \frac{d}{4}T^4 + \frac{e}{5}T^5)R_{gas} + \frac{h_{298}}{MW}]C_i \tag{58}$$

where h_{298} is the mass basis value of enthalpy at T= 298 K.

A.3.3 Entropy, s.

Entropy, for a thermally perfect gas, is given by the following expression (10:51):

$$\begin{aligned} s &= \int_{t_o}^{t} c_p \frac{dt}{t} - R\ln\frac{P}{P_o} \\ &= \phi - R\ln\frac{P}{P_o} \end{aligned} \tag{59}$$

Again, substituting eq. (56) into the integral portion of this equation yields the least squares coefficient form of the entropy parameter, ϕ_i, for the ith species (4:32):

$$\phi_i = (\phi_o + a\ln T + bT + \frac{c}{2}T^2 + \frac{d}{3}T^3 + \frac{e}{4}T^4)R_{gas}C_i \tag{60}$$

Appendix B and Appendix D show how these expressions are used to implement the imperfect gas model.

A.4 Code Modifications

The original code consisted of 46 subroutines. Three were added and eight were modified. The following is a summary of the altered subroutines. Appendix E contains the three additional subroutines and imperfect gas oblique shock wave solver.

Subroutine INITEM: initializes the arrays and variables. Values of coefficients from CET89 output are listed here.

Subroutine INPUT: read statements for input deck. Mass fractions are read in here.

Subroutine INVALU: initial value line for inlet to nozzle used to begin Riemann problem. Speed of sound and density require $\gamma(T)$ and gas constant, R_{gas}, for mixture.

Subroutine OUTINP: outputs the input deck.

Subroutine RMCONT: determines which Riemann solver to use. Updated z coefficient requires $\gamma(T)$.

Subroutine RMAPAP: solution of the Riemann problem.

Subroutine UPWIND1: marches the solution in the x direction.

Subroutine BNDWAV: marches the solution in x direction along boundaries.

Appendix B: Solution of the Riemann Problem

This discussion of the linearized-approximate method applied to the solution of a perfect gas Riemann problem is summarized from Doty's research (3). The reader is urged to consult reference (3) for more complete details.

As previously mentioned, the linearized-approximate method treats waves as isentropic expansions and compressions. By linearizing the Prandtl-Meyer equations, a closed-form equation results that requires no iteration. Figure 21 illustrates the setup of the Riemann problem.

B.1 Linearized-Approximate, Perfect Gas

For isentropic, planar, steady flow, the differential form of the compatibility relations, valid along Mach lines, is:

$$\sqrt{M^2-1}\; dP \pm \rho V^2 d\theta = 0 \qquad (61)$$

where the velocity magnitude and flow angle are:

$$V^2 = u^2+v^2 \qquad (62)$$

$$\theta = \tan^{-1}(v/u) \qquad (63)$$

Through substitution, manipulation, and realizing that $\rho=\gamma P/a^2$, eq. (61) is recast as:

$$\frac{dP}{P} \pm \frac{(\gamma u^2/a^2)}{\sqrt{M^2-1}}\, d(v/u) = 0 \qquad (64)$$

Two definitions and a relationship are introduced:

$$z \equiv \frac{(\gamma u^2/a^2)}{\sqrt{M^2-1}} \tag{65}$$

$$\sigma \equiv v/u \tag{66}$$

$$d[\ln(P)] = \frac{dP}{P} \tag{67}$$

Eq. (61) is then written more compactly as:

$$d[\ln(P)] \pm (z)d\sigma = 0 \tag{68}$$

Linearizing this yields:

$$\Delta[\ln(P)] \pm (z)\Delta\sigma = 0 \tag{69}$$

Referring to Figure 21, a positive wave, wave (3), is required to pass information from region (0) to region (2). Using the positive sign from eq. (69) yields:

$$([\ln(P)]_2-[\ln(P)]_0) + (z_0)(\sigma_2-\sigma_0) = 0 \tag{70}$$

Rearranging eq. (70) yields:

$$[\ln(P)]_2+(z_0)\sigma_2 = [\ln(P)]_0+(z_0)\sigma_0 \tag{71}$$

A similar process is performed for regions (6) and (4):

$$[\ln(P)]_4+(z_6)\sigma_4 = [\ln(P)]_6+(z_6)\sigma_6 \tag{72}$$

Recall that the pressure and flow slope must match across the contact surface:

$$\sigma_4 = \sigma_2 \tag{73}$$

$$P_4 = P_2 \tag{74}$$

Eqs. (71), (72), (73), and (74) represent four equations and four unknowns and may be solved in closed form. Substituting eqs. (73) and (74) into eq. (71) and rearranging gives:

$$[\ln(P)]_4 + (z_0)\sigma_4 = [\ln(P)]_0 + (z_0)\sigma_0 \tag{75}$$

Solving for σ_4 yields:

$$\sigma_4 = \frac{[\ln(P)]_0 - [\ln(P)]_6 + (z_6)\sigma_6 + (z_0)\sigma_0}{(z_6 + z_0)} \tag{76}$$

Using this value in the solution of P_4, in eq. (72), gives:

$$P_4 = \exp([\ln(P)]_6 + z_6(\sigma_4 - \sigma_6)) \tag{77}$$

Isentropic relation~ can be used to obtain density and speed of sound in regions (2) and (4) (only region (2) is shown here):

$$\frac{\rho_2}{\rho_0} = \left[\frac{P_2}{P_0}\right]^{1/\gamma} \tag{78}$$

$$a_2 = [\gamma P_2/\rho_2]^{1/2} \tag{79}$$

Conservation of stagnation enthalpy across a wave is used to obtain velocity (recall that $h=c_pT$):

$$\frac{h_{t_4}}{h_{t_6}} = \frac{c_p T_{t_4}}{c_p T_{t_6}} = \frac{T_4\left[1+\frac{\gamma-1}{2}M_4^2\right]}{T_6\left[1+\frac{\gamma-1}{2}M_6^2\right]} = 1 \qquad (80)$$

Eq. (80) can be solved explicitly for M_4; and the velocity components, in terms of flow slope, σ_4, are:

$$u_4 = M_4 a_4 \cos[\tan^{-1}(\sigma_4)] \qquad (81)$$

$$v_4 = M_4 a_4 \sin[\tan^{-1}(\sigma_4)] \qquad (82)$$

A similar process is done for region (2).

B.2 Linearized-approximate, Imperfect Gas

Pressure in region (4), P_4, from eq. (77), is also valid for an imperfect gas. Since pressure is the only property known in region (4), a relation is needed that does not rely on a calorically perfect form of conservation of stagnation enthalpy.

To determine temperature, relations are used that take advantage of the isentropic nature of the flow. For a thermally perfect gas, the differential quantity of entropy in a system is (10:45):

$$ds = c_p\frac{dt}{t} - R\frac{dP}{P} \qquad (83)$$

Integrating this yields (10:58):

$$s = \int_{t_o}^{t} c_p\frac{dt}{t} - R\ln\frac{P}{P_o} = \phi - R\ln\frac{P}{P_o} \qquad (84)$$

where the entropy parameter, ϕ, is defined as (10:58):

$$\phi = \int_{t_o}^{t} c_p \frac{dt}{t} \qquad (85)$$

Substituting eq. (56), on a molar basis, into eq. (85) yields the already integrated equation (10:58):

$$\phi = \left(\phi_o + a\ln t + bt + \frac{ct^2}{2} + \frac{dt^3}{3} + \frac{et^4}{4}\right) R \qquad (86)$$

Appendix A shows how the entropy parameter, ϕ, can be determined. A change in entropy is found from eq. (84) and written for regions (4) and (6) as (10:58):

$$s_4 - s_6 = \phi_4 - \phi_6 - R\ln\left(\frac{P_4}{P_6}\right) \qquad (87)$$

For isentropic flow, $\Delta s=0$ and eq. (87) is rewritten as:

$$\phi_4 = \phi_6 + R\ln\left(\frac{P_4}{P_6}\right) \qquad (88)$$

where ϕ_6 is found using eq. (86) in terms of the known temperature in region (6). The entropy parameter in region (4), ϕ_4, is the only unknown is this equation. Using eq. (86), temperature in region (4), T_4, is solved for iteratively using the secant method. Subroutine ENTROPY performs this operation and can be found in Appendix E. With temperature and pressure in region (4), density can be determined using the thermal equation of state.

As in the perfect gas case, conservation of stagnation enthalpy is used to find the velocity components, but in a different form. Stagnation enthalpy is determined in region (4) as:

$$h_{t_4} = h_6 + \frac{1}{2}(u^2+v^2)_6 \tag{89}$$

where static enthalpy, h_6, is a function of T_6 and is found using the methods described in Appendix A. Across wave (3), stagnation enthalpy in region (4) is found using the conservation of energy relationship, $h_{t6} = h_{t4}$. Subroutine ENTHGAM performs the calculation of enthalpy and ratio of specific heats based on temperature and can be found in Appendix E. Recalling that $\sigma=v/u$, stagnation enthalpy can be written as:

$$h_{t_4} = h_4 + \frac{1}{2}u_4^2(1+\sigma_4^2) \tag{90}$$

Solving for u_4 gives:

$$u_4 = \sqrt{\frac{2(h_{t_4}-h_4)}{1+\sigma_4^2}} \tag{91}$$

The y component of velocity is then computed as:

$$v_4 = u_4\sigma_4 \tag{92}$$

Mach number can now be computed:

$$M_4 = \frac{V_{mag}}{a_4} \tag{93}$$

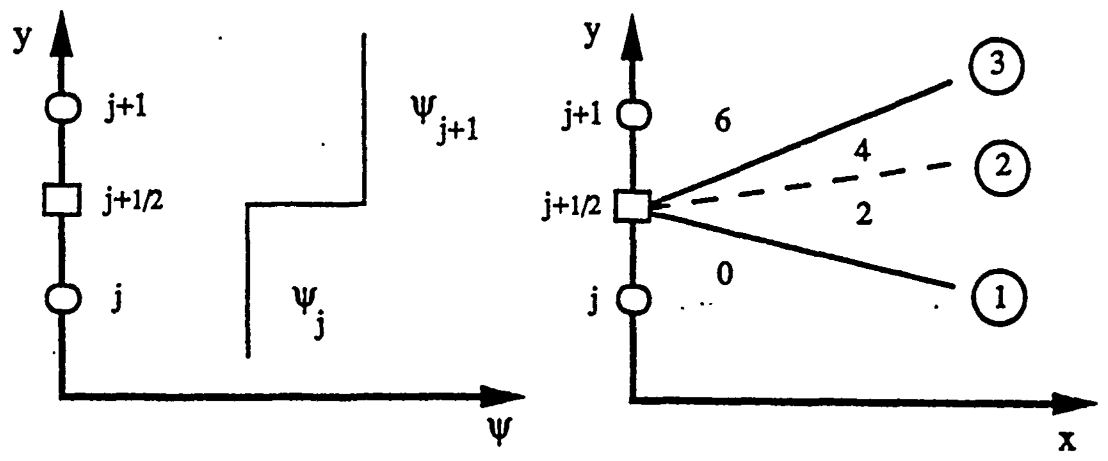

Figure 21 Riemann problem in solution space and wave pattern (3:174)

Appendix C: Numerical Algorithm

C.1 Introduction

The methodology of flux-difference-splitting for supersonic planar flow and the first-order accurate marching scheme is presented in this appendix. This is a summary of the numerical schemes and the reader is referred to reference (3) for further details.

C.2 Riemann Fluxes

After solving the Riemann problem by one of the three methods described, the Riemann fluxes are computed. Figure 22 shows a schematic of the flux differencing and splitting. The solution of the Riemann problem in Appendix B is the basis for computing the fluxes.

To illustrate, consider the flux vector, E1, for the Riemann regions 0, 2, 4, 6:

$$(E1)_0 = \rho_0 u_0 \tag{94}$$

$$(E1)_2 = \rho_2 u_2 \tag{95}$$

$$(E1)_4 = \rho_4 u_4 \tag{96}$$

$$(E1)_6 = \rho_6 u_6 \tag{97}$$

The Riemann flux-differences are the differences of the E1 vector taken across waves (1), (2), and (3):

$$(dE1)_{wave3} = (E1)_6 - (E1)_4 \tag{98}$$

$$(dE1)_{wave2} = (E1)_4 - (E1)_2 \tag{99}$$

$$(dE1)_{wave1} = (E1)_2 - (E1)_0 \tag{100}$$

The sum of the contributions across all three waves gives the total contribution at Riemann node j+1/2 (3:184):

$$(dE1)_{j+1/2} = [(dE1)_{wave3} + (dE1)_{wave2} + (dE1)_{wave1}]_{j+1/2} \tag{101}$$

Riemann node j-1/2 is handled in a similar fashion, as are the other E and F vector components.

C.3 Splitting the Flux Differences

With the Riemann problem solved and the flux differences taken, the differences are now split to determine which information from Riemann nodes j+1/2 and j-1/2 will reach node j at the next plane, i+1 (3:183). Figure 22 illustrates the flux difference and splitting.

Streamlines and characteristics determine which direction to send the differenced fluxes. As described by Doty from Chakravarthy's report, Godunov and Osher developed identical methods for splitting the flux differences, in unsteady flow, based on the sign of the slopes of the waves emanating from the Riemann node (3:184). This concept was extended to steady, two-dimensional flow. For example, at node j-1/2, if the slope of waves (1), (2), or (3) is positive, the flux difference of the E and F vectors across that wave(s) contributes to the solution at node i+1,j (3:185). A similar process occurs at Riemann node j+1/2 where this time only flux differences with a negative slope contribute to the solution at i+1,j. In equation form, summing the positive and negative contributions at the Riemann nodes j+1/2 and j-1/2, respectively, yields (3:186):

$$dE^{-}_{j+1/2} = \left[dE^{wave3}_{j+1/2}\right]^{-} + \left[dE^{wave2}_{j+1/2}\right]^{-} + \left[dE^{wave1}_{j+1/2}\right]^{-} \tag{102}$$

$$dE^{+}_{j-1/2} = \left[dE^{wave3}_{j-1/2}\right]^{+} + \left[dE^{wave2}_{j-1/2}\right]^{+} + \left[dE^{wave1}_{j-1/2}\right]^{+} \tag{103}$$

where the positive or negative signs denote positively or negatively sloped flux differences. All, none, or some of the waves can contribute to the solution, depending on their slopes. The same procedure is applied to the F vector. For further details, refer to Appendix J of reference (3).

C.4 First-Order Accurate Marching Scheme

The governing equations for supersonic flow are hyperbolic and can thus be solved by a marching technique. The transformed governing equations are developed in Appendix D of reference (3) and are repeated here without proof (3:188):

$$\frac{\partial(\boldsymbol{E})}{\partial\zeta} = \eta_x \frac{\partial(\boldsymbol{E})}{\partial\eta} - \eta_y \frac{\partial(\boldsymbol{F})}{\partial\eta} \tag{104}$$

In order to march the flow in the x or ζ direction a combination of finite difference and flux difference approximations are used for the partial derivatives of E and F in eq. (104). Substituting these approximations directly into eq. (104) yields:

$$\frac{\Delta_i(E)}{\Delta\zeta} = -\eta_x\frac{\Delta_j(E)}{\Delta\eta} - \eta_y\frac{\Delta_j(F)}{\Delta\eta} \qquad (105)$$

where Δ_i is the finite difference operator for the ζ direction and Δ_j is the flux difference operator for the η direction (3:188). For convenience, $\Delta\eta$ is chosen to be unity. Rearranging yields:

$$\Delta_i(E) = -\Delta\zeta\eta_x\Delta_j(E) - \Delta\zeta\eta_y\Delta_j(F) \qquad (106)$$

The solution is advanced in the x direction using a finite difference operator for $\Delta_i(E)$ in eq. (106):

$$\Delta_i(E) = E_j^{i+1} - E_j^i \qquad (107)$$

Substituting this into eq. (106) and rearranging yields (3:189):

$$E_j^{i+1} = E_j^i - \Delta\zeta\,\eta_x\Delta_j(E) - \Delta\zeta\,\eta_y\Delta_j(F) \qquad (108)$$

As discussed earlier, the solution uses negatively biased information from Riemann node j+1/2 and positively biased information from Riemann node j-1/2. Recalling the results of eqs. (102) and (103), the first-order accurate FDS representations for $\Delta_j E$ and $\Delta_j F$ operators are (3:190):

$$\Delta_j(E) = \{dE^+_{j-1/2} + dE^-_{j+1/2}\} \qquad (109)$$

$$\Delta_j(\boldsymbol{F}) = \{d\boldsymbol{F}^{+}_{j-1/2} + d\boldsymbol{F}^{-}_{j+1/2}\} \tag{110}$$

Substituting eqs. (109) and (110) into eq. (108) yields the first-order accurate FDS approximation to the solution of the governing equations (3:190):

$$\boldsymbol{E}^{i+1}_j = \boldsymbol{E}^{i}_j - \Delta\zeta\, \eta_x\left[d\boldsymbol{E}^{+}_{j-1/2} + d\boldsymbol{E}^{-}_{j+1/2}\right] - \Delta\zeta\, \eta_y\left[d\boldsymbol{F}^{+}_{j-1/2} + d\boldsymbol{F}^{-}_{j+1/2}\right] \tag{111}$$

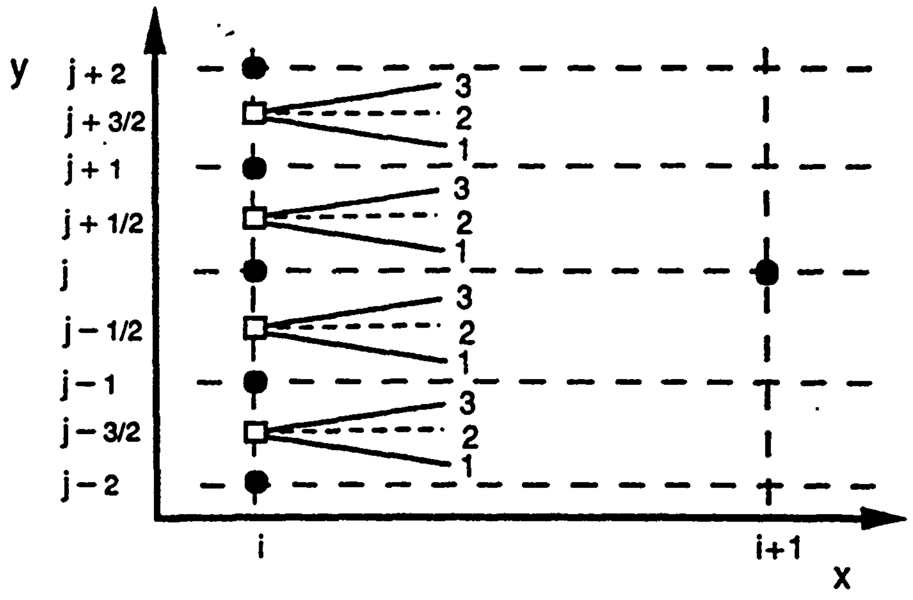

Figure 22 Flux differences and splitting (3:187)

Appendix D: Decoding the Solution

D.1 Introduction

After marching the solution to the next plane, i+1, the newly calculated **E** and **F** vectors must be decoded into primitive variables so the Riemann problem may be solved. This will then complete the "loop" needed to solve the flow field from one plane to next.

D.2 Decoding, Perfect Gas

Decoding the **E** vector requires that all components be solved simultaneously. The **E** vector with its four components are known from marching to the i+1 plane. It is repeated here for convenience:

$$\boldsymbol{E} = \begin{bmatrix} \rho u \\ \rho u^2+P \\ \rho uv \\ u(\rho e+P) \end{bmatrix} \tag{112}$$

The ρe term in the E4 component can be written as:

$$\rho e = \rho \hat{u} + \frac{1}{2}\rho(u^2+v^2) \tag{113}$$

For a perfect gas, specific internal energy, $\hat{u}$, can be written as:

$$\hat{u} = c_v T \tag{114}$$

The following relation holds for a thermally perfect gas

(10:44):

$$C_v = \frac{R_{gas}}{\gamma - 1} \tag{115}$$

Recalling the thermal equation of state, substituting eq. (115) into eq. (114), and substituting that result into eq. (113) yields:

$$\rho e = \frac{P}{\gamma - 1} + \frac{1}{2}\rho(u^2+v^2) \tag{116}$$

This is the fifth equation to close the system of five unknowns. Eq. (116) can be substituted into the E4 component to give:

$$E4 = u[(\frac{P}{\gamma-1}+\frac{1}{2}\rho(u^2+v^2))+P] \tag{117}$$

Expanding and rearranging eq. (117) gives:

$$E4 = uP\left[\frac{\gamma}{\gamma-1}\right]+\frac{1}{2}\rho u(u^2+v^2) \tag{118}$$

Recalling that E1= ρu, E2, E3, and E4 can be written as:

$$E2 = (E1)u+P \tag{119}$$

$$E3 = (E1)v \tag{120}$$

$$E4 = uP\left[\frac{\gamma}{\gamma-1}\right]+\frac{1}{2}(E1)(u^2)+\frac{1}{2}(E1)(v^2) \tag{121}$$

Solving eq. (120) for the y component of velocity, v, gives:

$$v = \frac{E3}{E1} \tag{122}$$

Eq. (119) can be solved for P:

$$P = E2-(E1)u \tag{123}$$

Eqs. (122) and (123) can be substituted into eq. (121) to yield:

$$E4 = u\left[\frac{\gamma}{\gamma-1}\right][(E2)-(E1)u]+\frac{1}{2}(E1)(u^2)+\frac{1}{2}(E1)\left[\frac{E3}{E1}\right]^2 \tag{124}$$

These manipulations have resulted in an E4 component that is a function of the known **E** vector components and the one unknown, u, and can now be cast as a quadratic equation:

$$\left[\frac{\gamma+1}{2(\gamma-1)}(E1)\right]u^2-\left[\frac{\gamma}{\gamma-1}(E2)\right]u+\left[(E4)-\frac{1}{2}\frac{(E3)^2}{(E1)}\right] = 0 \tag{125}$$

$$au^2+bu+c=0 \tag{126}$$

This quadratic equation can now be used to find the x component of velocity, u, where it has been determined that the positive value is the correct root (3:222). The primitive variables are:

$$u = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{127}$$

$$v = \frac{E3}{E1} \tag{128}$$

$$P = E2-(E1)u \tag{129}$$

$$\rho = \frac{(E1)}{u} \tag{130}$$

$$\rho e = \frac{P}{\gamma-1}+\frac{1}{2}\rho(u^2+v^2) \tag{131}$$

This completes the decoding process for a perfect gas.

D.3 Decoding, Imperfect Gas

Decoding the E vector for an imperfect gas requires that the ρe term of the E4 component be consistent with the imperfect gas assumption. Recall eq. (113) , rewritten here for convenience:

$$\rho e = \rho\hat{u} + \frac{1}{2}\rho(u^2+v^2) \tag{132}$$

Specific internal energy, $\hat{u}$, for an imperfect gas is given as (10:58):

$$\hat{u} = h - RT \tag{133}$$

Substituting eq. (133) into eq. (132) yields the thermally perfect form of total internal energy:

$$\rho e = \rho h - P + \rho \frac{1}{2}(u^2+v^2) \tag{134}$$

Substituting eq. (134) into the E4 component and canceling the pressure terms gives:

$$E4 = u[(\rho h - P + \frac{1}{2}\rho(u^2+v^2) + P] \tag{135}$$

$$E4 = \rho u h + \frac{1}{2}\rho u(u^2+v^2) \tag{136}$$

Substituting the E1 component into eq. (136) yields the final form of the E4 component:

$$E4 = h(E1) + \frac{1}{2}(E1)u^2 + \frac{1}{2}\frac{(E3)^2}{E1} \tag{137}$$

This equation is now a function of the x component of velocity, u, and enthalpy, which itself is a function of temperature from the least squares coefficient. An expression for the velocity component, u, must be found that is a function of temperature so as to obtain an equation for E4 that will, subsequently, also have one unknown, temperature. Start by solving the E1 component for u: u= E1/ρ. Substitute the thermal equation of state for ρ:

$$u = \frac{(E1)RT}{P} \tag{138}$$

Solve the E2 component for P and substitute the result into eq. (138):

$$u = \frac{(E1)RT}{E2-(E1)u} \tag{139}$$

Rearrange eq. (139) into a quadratic equation in the x component of velocity, u:

$$(E1)u^2 + (E2)u + (E1)RT = 0 \tag{140}$$

$$u = \frac{E2 \pm \sqrt{(E2)-4(E1)((E1)RT)}}{2(E1)} \tag{141}$$

From this manipulation, it is seen that u is now a function of temperature. As was desired, the E4 component is now also a function of only temperature. Temperature can now be found by an iterative technique, such as the secant method. With temperature determined, u can be obtained by the quadratic equation. Similar to the perfect gas case, pressure can be obtained from eq. (129) and density is obtained from eq. (130).

The decoding process for the imperfect gas model is now complete and accounts the variable specific heats as a function of temperature.

Appendix E: New Subroutines

This appendix contains the three additional subroutines ENTROPY, ENTHGAM, and IMPERFECT, and the code for solving the imperfect gas oblique shock wave problem. Modifications to the complete code are denoted by * in FORTRAN column one.

```
      subroutine entropy (p1,p2,tt,t)
c
c     computes temperature after an isentropic wave based
c     on the pressure in the regions before and after wave
c     and temperature in the region prior to the wave.
c
      common/coefficient/
     1   alN2,blN2,clN2,dlN2,elN2,plN2,hlN2,
     2   ahN2,bhN2,chN2,dhN2,ehN2,phN2,hhN2,
     3   alO2,blO2,clO2,dlO2,elO2,plO2,hlO2,
     4   ahO2,bhO2,chO2,dhO2,ehO2,phO2,hhO2,
     5   alAr,plAr,hlAr,
     6   ahAr,phAr,hhAr,
     7   alH,blH,clH,dlH,elH,plH,hlH,
     8   ahH,bhH,chH,dhH,ehH,phH,hhH,
     9   alOH,blOH,clOH,dlOH,elOH,plOH,hlOH,
     9   ahOH,bhOH,chOH,dhOH,ehOH,phOH,hhOH,
     1   alH2,blH2,clH2,dlH2,elH2,plH2,hlH2,
     2   ahH2,bhH2,chH2,dhH2,ehH2,phH2,hhH2,
     3   alNO,blNO,clNO,dlNO,elNO,plNO,hlNO,
     4   ahNO,bhNO,chNO,dhNO,ehNO,phNO,hhNO,
     5   alO,blO,clO,dlO,elO,plO,hlO,
     6   ahO,bhO,chO,dhO,ehO,phO,hhO,
     7   alH2O,blH2O,clH2O,dlH2O,elH2O,plH2O,hlH2O,
     8   ahH2O,bhH2O,chH2O,dhH2O,ehH2O,phH2O,hhH2O,
     9   lN2,lO2,lAr,lH,lOH,lH2,lNO,lO,lH2O

      common/species/
     1   xmsfN2,xmwN2,xmsfO2,xmwO2,xmsfAr,xmwAr,
     2   xmsfH,xmwH,xmsfOH,xmwOH,xmsfH2,xmwH2,xmsfNO,xmwNO,
     3   xmsfO,xmwO,xmsfH2O,xmwH2O,
     4   runiv,rmix
c-----------------------------------------------------------------
      iter=0
      itmax=50
      rmtol=.000001
      t1=tt

      if(t1.le.1000.) then
        phimix1=
     1    lN2*(plN2+alN2*log(t1)+blN2*t1+clN2*t1*t1/2.+dlN2*
     2     t1**3./3.+elN2*t1**4./4.)*runiv*xmsfN2/xmwN2 +
     3    lO2*(plO2+alO2*log(t1)+blO2*t1+clO2*t1*t1/2.+dlO2*
     4     t1**3./3.+elO2*t1**4./4.)*runiv*xmsfO2/xmwO2 +
     5    lAr*(plAr+alAr*log(t1))*runiv*xmsfAr/xmwAr+
     6    lH*(plH+alH*log(t1))*runiv*xmsfH/xmwH+
     7    lOH*(plOH+alOH*log(t1)+blOH*t1+clOH*t1*t1/2.+dlOH*
     8     t1**3./3.+elOH*t1**4./4.)*runiv*xmsfOH/xmwOH +
9      1H2*(p1H2+a1H2*log(t1)+b1H2*t1+c1H2*t1*t1/2.+d1H2*
9       t1**3./3.+e1H2*t1**4./4.)*runiv*xmsfH2/xmwH2 +
1      1NO*(p1NO+a1NO*log(t1)+b1NO*t1+c1NO*t1*t1/2.+d1NO*
2       t1**3./3.+e1NO*t1**4./4.)*runiv*xmsfNO/xmwNO +
3      1O*(p1O+a1O*log(t1)+b1O*t1+c1O*t1*t1/2.+d1O*
4       t1**3./3.+e1O*t1**4./4.)*runiv*xmsfO/xmwO +
5      1H2O*(p1H2O+a1H2O*log(t1)+b1H2O*t1+c1H2O*t1*t1/2.
6       +d1H2O*t1**3./3.+e1H2O*t1**4./4.)
7       *runiv*xmsfH2O/xmwH2O
  else
    phimix1=
1      1N2*(phN2+ahN2*log(t1)+bhN2*t1+chN2*t1*t1/2.+dhN2*
2       t1**3./3.+ehN2*t1**4./4.)*runiv*xmsfN2/xmwN2 +
3     1O2*(phO2+ahO2*log(t1)+bhO2*t1+chO2*t1*t1/2.+dhO2*
4      t1**3./3.+ehO2*t1**4./4.)*runiv*xmsfO2/xmwO2 +
5     1Ar*(phAr+ahAr*log(t1))*runiv*xmsfAr/xmwAr +
6     1H*(phH+ahH*log(t1))*runiv*xmsfH/xmwH +
7     1OH*(phOH+ahOH*log(t1)+bhOH*t1+chOH*t1*t1/2.+dhOH*
8      t1**3./3.+ehOH*t1**4./4.)*runiv*xmsfOH/xmwOH +
9     1H2*(phH2+ahH2*log(t1)+bhH2*t1+chH2*t1*t1/2.+dhH2*
9      t1**3./3.+ehH2*t1**4./4.)*runiv*xmsfH2/xmwH2 +
1     1NO*(phNO+ahNO*log(t1)+bhNO*t1+chNO*t1*t1/2.+dhNO*
2      t1**3./3.+ehNO*t1**4./4.)*runiv*xmsfNO/xmwNO +
3     1O*(phO+ahO*log(t1)+bhO*t1+chO*t1*t1/2.+dhO*
4      t1**3./3.+ehO*t1**4./4.)*runiv*xmsfO/xmwO +
5     1H2O*(phH2O+ahH2O*log(t1)+bhH2O*t1+chH2O*t1*t1/2.
6      +dhH2O*t1**3./3.+ehH2O*t1**4./4.)
7      *univ*xmsfH2O/xmwH2O
  end if

  phimix2=phimix1+rmix*log(p2/p1)
  t2=t1+50.

  do 100 iter=1,itmax
  if(t2.le.1000.) then
    phimix2g=
1     1N2*(p1N2+a1N2*log(t2)+b1N2*t2+c1N2*t2*t2/2.+d1N2*
2      t2**3./3.+e1N2*t2**4./4.)*runiv*xmsfN2/xmwN2 +
3    1O2*(p1O2+a1O2*log(t2)+b1O2*t2+c1O2*t2*t2/2.+d1O2*
4     t2**3./3.+e1O2*t2**4./4.)*runiv*xmsfO2/xmwO2 +
5    1Ar*(p1Ar+a1Ar*log(t2))*runiv*xmsfAr/xmwAr +
6    1H*(p1H+a1H*log(t2))*runiv*xmsfH/xmwH +
7    1OH*(p1OH+a1OH*log(t2)+b1OH*t2+c1OH*t2*t2/2.+d1OH*
8     t2**3./3.+e1OH*t2**4./4.)*runiv*xmsfOH/xmwOH +
9    1H2*(p1H2+a1H2*log(t2)+b1H2*t2+c1H2*t2*t2/2.+d1H2*
9     t2**3./3.+e1H2*t2**4./4.)*runiv*xmsfH2/xmwH2 +
1    1NO*(p1NO+a1NO*log(t2)+b1NO*t2+c1NO*t2*t2/2.+d1NO*
2     t2**3./3.+e1NO*t2**4./4.)*runiv*xmsfNO/xmwNO +
3    1O*(p1O+a1O*log(t2)+b1O*t2+c1O*t2*t2/2.+d1O*
4     t2**3./3.+e1O*t2**4./4.)*runiv*xmsfO/xmwO +
     5  lH2O*(plH2O+alH2O*log(t2)+blH2O*t2+clH2O*t2*t2/2.
     6   +dlH2O*t2**3./3.+elH2O*t2**4./4.)
     7   *runiv*xmsfH2O/xmwH2O
       else
         phimix2g=
     1  lN2*(phN2+ahN2*log(t2)+bhN2*t2+chN2*t2*t2/2.+dhN2*
     2   t2**3./3.+ehN2*t2**4./4.)*runiv*xmsfN2/xmwN2 +
     3  lO2*(phO2+ahO2*log(t2)+bhO2*t2+chO2*t2*t2/2.+dhO2*
     4   t2**3./3.+ehO2*t2**4./4.)*runiv*xmsfO2/xmwO2 +
     5  lAr*(phAr+ahAr*log(t2))*runiv*xmsfAr/xmwAr +
     6  lH*(phH+ahH*log(t2))*runiv*xmsfH/xmwH +
     7  lOH*(phOH+ahOH*log(t2)+bhOH*t2+chOH*t2*t2/2.+dhOH*
     8   t2**3./3.+ehOH*t2**4./4.)*runiv*xmsfOH/xmwOH +
     9  lH2*(phH2+ahH2*log(t2)+bhH2*t2+chH2*t2*t2/2.+dhH2*
     9   t2**3./3.+ehH2*t2**4./4.)*runiv*xmsfH2/xmwH2 +
     1  lNO*(phNO+ahNO*log(t2)+bhNO*t2+chNO*t2*t2/2.+dhNO*
     2   t2**3./3.+ehNO*t2**4./4.)*runiv*xmsfNO/xmwNO +
     3  lO*(phO+ahO*log(t2)+bhO*t2+chO*t2*t2/2.+dhO*
     4   t2**3./3.+ehO*t2**4./4.)*runiv*xmsfO/xmwO +
     5  lH2O*(phH2O+ahH2O*log(t2)+bhH2O*t2+chH2O*t2*t2/2.
     6   +dhH2O*t2**3./3.+ehH2O*t2**4./4.)
     7   *runiv*xmsfH2O/xmwH2O
       end if

        f1=phimix2-phimix2g
        slope=(phimix2g-phimix1)/(t2-t1)
        t3=t2+f1/slope
        if(abs((t3-t2)/t2).lt.rmtol) go to 20
        t1=t2
        t2=t3
        phimix1=phimix2g
        iter=iter+1
  100   continue
        write(6,*)'temperature for region after shock (derived
      2 from entropy) failed to converge. Iters=',iter
        stop
   20   continue
        t=t3
        return
        end
c================end of subroutine entropy================
      subroutine enthgam(t,hmix,gmix)

c
c    computes enthalpy and gamma as a function of
c    temperature based on coefficients from CET85 program
c    (JANAF tables curve fitted)
c
       common/coefficient/
     1     alN2,blN2,clN2,dlN2,elN2,plN2,hlN2,
     2     ahN2,bhN2,chN2,dhN2,ehN2,phN2,hhN2,
     3     alO2,blO2,clO2,dlO2,elO2,plO2,hlO2,
     4     ahO2,bhO2,chO2,dhO2,ehO2,phO2,hhO2,
     5     alAr,plAr,hlAr,
     6     ahAr,phAr,hhAr,
     7     alH,blH,clH,dlH,elH,plH,hlH,
     8     ahH,bhH,chH,dhH,ehH,phH,hhH,
     9     alOH,blOH,clOH,dlOH,elOH,plOH,hlOH,
     9     ahOH,bhOH,chOH,dhOH,ehOH,phOH,hhOH,
     1     alH2,blH2,clH2,dlH2,elH2,plH2,hlH2,
     2     ahH2,bhH2,chH2,dhH2,ehH2,phH2,hhH2,
     3     alNO,blNO,clNO,dlNO,elNO,plNO,hlNO,
     4     ahNO,bhNO,chNO,dhNO,ehNO,phNO,hhNO,
     5     alO,blO,clO,dlO,elO,plO,hlO,
     6     ahO,bhO,chO,dhO,ehO,phO,hhO,
     7     alH2O,blH2O,clH2O,dlH2O,elH2O,plH2O,hlH2O,
     8     ahH2O,bhH2O,chH2O,dhH2O,ehH2O,phH2O,hhH2O,
     9     lN2,lO2,lAr,lH,lOH,lH2,lNO,lO,lH2O

       common/species/
     1    xmsfN2,xmwN2,xmsfO2,xmwO2,xmsfAr,xmwAr,
     2    xmsfH,xmwH,xmsfOH,xmwOH,xmsfH2,xmwH2,xmsfNO,xmwNO,
     3    xmsfO,xmwO,xmsfH2O,xmwH2O,
     4    runiv,rmix
c-----------------------------------------------------
c  h and cp coefficients from cet85
c
       if(t.le.1000.) then
         hcoefN2=hlN2+alN2*t+blN2*t*t/2.+clN2*t**3./3.+dlN2*
     2          t**4./4.+elN2*t**5./5.
         hcoefO2=hlO2+alO2*t+blO2*t*t/2.+clO2*t**3./3.+dlO2*
     2          t**4./4.+elO2*t**5./5.
         hcoefAr=hlAr+alAr*t
         hcoefH=hlH+alH*t
         hcoefOH=hlOH+alOH*t+blOH*t*t/2.+clOH*t**3./3.+dlOH*
     2          t**4./4.+elOH*t**5./5.
         hcoefH2=hlH2+alH2*t+blH2*t*t/2.+clH2*t**3./3.+dlH2*
     2          t**4./4.+elH2*t**5./5.
         hcoefNO=hlNO+alNO*t+blNO*t*t/2.+clNO*t**3./3.+dlNO*
     2          t**4./4.+elNO*t**5./5.
         hcoefO=hlO+alO*t+blO*t*t/2.+clO*t**3./3.+dlO*
     2          t**4./4.+elO*t**5./5.
        hcoefH2O=hlH2O+alH2O*t+blH2O*t*t/2.+clH2O*t**3./3.
     2           +dlH2O*t**4./4.+elH2O*t**5./5.
c
c
        cpcoefN2=alN2+blN2*t+clN2*t*t+dlN2*t**3.+elN2*t**4.
        cpcoefO2=alO2+blO2*t+clO2*t*t+dlO2*t**3.+elO2*t**4.
        cpcoefAr=alAr
        cpcoefH=alH
        cpcoefOH=alOH+blOH*t+clOH*t*t+dlOH*t**3.+elOH*t**4.
        cpcoefH2=alH2+blH2*t+clH2*t*t+dlH2*t**3.+elH2*t**4.
        cpcoefNO=alNO+blNO*t+clNO*t*t+dlNO*t**3.+elNO*t**4.
        cpcoefO=alO+blO*t+clO*t*t+dlO*t**3.+elO*t**4.
        cpcoefH2O=alH2O+blH2O*t+clH2O*t*t+dlH2O*t**3.
       2         +elH2O*t**4.
c
      else
c
        hcoefN2=hhN2+ahN2*t+bhN2*t*t/2.+chN2*t**3./3.+dhN2*
     2          t**4./4.+ehN2*t**5./5.
        hcoefO2=hhO2+ahO2*t+bhO2*t*t/2.+chO2*t**3./3.+dhO2*
     2          t**4./4.+ehO2*t**5./5.
        hcoefAr=hhAr+ahAr*t
        hcoefH=hhH+ahH*t
        hcoefOH=hhOH+ahOH*t+bhOH*t*t/2.+chOH*t**3./3.+dhOH*
     2          t**4./4.+ehOH*t**5./5.
        hcoefH2=hhH2+ahH2*t+bhH2*t*t/2.+chH2*t**3./3.+dhH2*
     2          t**4./4.+ehH2*t**5./5.
        hcoefNO=hhNO+ahNO*t+bhNO*t*t/2.+chNO*t**3./3.+dhNO*
     2          t**4./4.+ehNO*t**5./5.
        hcoefO=hhO+ahO*t+bhO*t*t/2.+chO*t**3./3.+dhO*
     2          t**4./4.+ehO*t**5./5.
        hcoefH2O=hhH2O+ahH2O*t+bhH2O*t*t/2.+chH2O*t**3./3.
     2            +dhH2O*t**4./4.+ehH2O*t**5./5.
c
        cpcoefN2=ahN2+bhN2*t+chN2*t*t+dhN2*t**3.+ehN2*t**4.
        cpcoefO2=ahO2+bhO2*t+chO2*t*t+dhO2*t**3.+ehO2*t**4.
        cpcoefAr=ahAr
        cpcoefH=ahH
        cpcoefOH=ahOH+bhOH*t+chOH*t*t+dhOH*t**3.+ehOH*t**4.
        cpcoefH2=ahH2+bhH2*t+chH2*t*t+dhH2*t**3.+ehH2*t**4.
        cpcoefNO=ahNO+bhNO*t+chNO*t*t+dhNO*t**3.+ehNO*t**4.
        cpcoefO=ahO+bhO*t+chO*t*t+dhO*t**3.+ehO*t**4.
        cpcoefH2O=ahH2O+bhH2O*t+chH2O*t*t+dhH2O*t**3.
     2  +ehH2O*t**4.
c
       end if
c
c  Compute h for each species.  Molar values of h (t=0K)
c     must be found for each species. These can be found
```

```
c     in the JANAF tables.
c
      hN2=hcoefN2*runiv+8669000.
      hO2=hcoefO2*runiv+8682000.
      hAr=hcoefAr*runiv+6196450.
      hH=hcoefH*runiv+6196504.
      hOH=hcoefOH*runiv+8815688.
      hH2=hcoefH2*runiv+8468416.
      hNO=hcoefNO*runiv+9192248.
      hO=hcoefO*runiv+6727872.
      hH2O=hcoefH2O*runiv+9904000.
c
c  Compute h of mixture
c
      hmix=lN2*hN2*xmsfN2/xmwN2+
     2     lO2*hO2*xmsfO2/xmwO2+
     3     lAr*hAr*xmsfAr/xmwAr+
     4     lH*hH*xmsfH/xmwH+
     5     lOH*hOH*xmsfOH/xmwOH+
     6     lH2*hH2*xmsfH2/xmwH2+
     7     lNO*hNO*xmsfNO/xmwNO+
     8     lO*hO*xmsfO/xmwO+
     9     lH2O*hH2O*xmsfH2O/xmwH2O
c
c  Compute cp of mixture
c
      cpmix=lN2*cpcoefN2*runiv*xmsfN2/xmwN2
     2     +lO2*cpcoefO2*runiv*xmsfO2/xmwO2
     3     +lAr*cpcoefAr*runiv*xmsfAr/xmwAr
     4     +lH*cpcoefH*runiv*xmsfH/xmwH
     5     +lOH*cpcoefOH*runiv*xmsfOH/xmwOH
     6     +lH2*cpcoefH2*runiv*xmsfH2/xmwH2
     7     +lNO*cpcoefNO*runiv*xmsfNO/xmwNO
     8     +lO*cpcoefO*runiv*xmsfO/xmwO
     9     +lH2O*cpcoefH2O*runiv*xmsfH2O/xmwH2O
c
c  Compute cv and gamma of the mixture
c
      cvmix=cpmix-rmix
      gmix=cpmix/cvmix
c
      return
      end
c=============== end of subroutine enthgam==============
      subroutine imperfect (j,g)

c
c     computes the primitive variables from the e
c     vector based on a thermally perfect gas
c
      parameter (jdim=501)

      common/species/
     1   xmsfN2,xmwN2,xmsfO2,xmwO2,xmsfAr,xmwAr,
     2   xmsfH,xmwH,xmsfOH,xmwOH,xmsfH2,xmwH2,xmsfNO,xmwNO,
     3   xmsfO,xmwO,xmsfH2O,xmwH2O,
     4   runiv,rmix

      common/solution/ cn,olddx,pamb,betafd,rmtol,xstop,
     2       phi,x(jdim,2),y(jdim,2),rh(jdim,2),u(jdim,2),
     3       v(jdim,2),p(jdim,2),t(jdim,2),rhe(jdim,2),
     4       xmach(jdim,2)

      common/riem/ rh0(jdim),rh2(jdim),rh4(jdim),rh6(jdim),
     2    u0(jdim),u2(jdim),u4(jdim),u6(jdim),xmach2(jdim),
     3    v0(jdim),v2(jdim),v4(jdim),v6(jdim),xmach4(jdim),
     4    p0(jdim),p2(jdim),p4(jdim),p6(jdim),
     5    rhe0(jdim),rhe2(jdim),rhe4(jdim),rhe6(jdim),
     6    slp10(jdim),slp22(jdim),slp36(jdim)

      common/efvector/
     1   e1(jdim,2),e2(jdim,2),e3(jdim,2),e4(jdim,2),
     2   f1(jdim,2),f2(jdim,2),f3(jdim,2),f4(jdim,2)

      common/compflg1/
     1   jmarch,ibnd,jattach,ioptim,itermax,iteropt,
     2   imax,jnozz,ix,knozzle,nwpnts,kucowl,klcowl,
     3   ipsarc,igrid,
     4   islpnoz,jucowl,jlcowl,ipackin,ipackex,
     5   jpackin,jpackex,
     6   iriem,irmbnd,inpnoz,ivint,ivext,nixint,
     7   nixext,kbot,
     5   method,limit,ipastuc,ipastlc,ixcowl,irmfix,
     9   itriem,itmoc,
     9   inpcwl,icompar,ktrk,idoext
c
c-----------------------------------------------------------------
c      j=1
       iter=0
       itmax=50
       if (j.eq.jnozz.or.j.eq.jlcowl) then
       t1=p0(j)/rh0(j)/rmix
       else
       t1=p6(j)/rh6(j)/rmix
       endif
       t2=t1+50..
       call enthgam (t1,h1,g1)
       ua=(e2(j,2)+sqrt(e2(j,2)*e2(j,2)-4.*e1(j,2)*
     2   e1(j,2)*rmix*t1))/2./e1(j,2)
       fa=h1*e1(j,2)+.5*e1(j,2)*ua*ua+.5*e3(j,2)*
     2   e3(j,2)/e1(j,2)

    10 call enthgam (t2,h2,g2)
       ub=(e2(j,2)+sqrt(e2(j,2)*e2(j,2)-4.*e1(j,2)*
     2   e1(j,2)*rmix*t2))/2./e1(j,2)
       fb=h2*e1(j,2)+.5*e1(j,2)*ub*ub+.5*e3(j,2)*
     2   e3(j,2)/e1(j,2)
       slope=(fb-fa)/(t2-t1)
       t3=t2+(e4(j,2)-fb)/slope
       if(abs((t3-t2)/t2).le.rmtol) go to 20
       if(iter.gt.itmax) then
       write(6,*)'sub. imperfect failed to converge, plane
     2           ix=',ix
       write(6,*)'and stops at node j=',j
       stop
       endif
       fa=fb
       t1=t2
       t2=t3
       iter=iter+1
       go to 10
    20 continue
       g=g2
       u(j,2)=(e2(j,2)+sqrt(e2(j,2)*e2(j,2)-4.*e1(j,2)*
     2       e1(j,2)*rmix*t3))/2./e1(j,2)
       v(j,2)=e3(j,2)/e1(j,2)
       if(abs(v(j,2)).lt.1.e-3) v(j,2)=0.0
       if(abs(e3(j,2)).lt.1.e-3) e3(j,2)=0.0
       p(j,2)=e2(j,2)-e1(j,2)*u(j,2)
       rh(j,2)=e1(j,2)/u(j,2)
       rhe(j,2)=rh(j,2)*h2-p(j,2)+(rh(j,2)/2.)*
     2        (u(j,2)*u(j,2)+v(j,2)*v(j,2))
       ax=asnd(g2,p(j,2),rh(j,2))
       vmag=sqrt(u(j,2)*u(j,2)+v(j,2)*v(j,2))
       xmach(j,2)=vmag/ax
       t(j,2)=t3
       if(ix.eq.39.and.(j.eq.50.or.j.eq.51))then
       endif

       return
       end
```

```
c=================================================================
      subroutine enthgam(t,hmix,gmix)
c
c     computes enthalpy and gamma as a function of
c     temperature based on coefficients from CET85 program
c     (JANAF tables curve fitted)
c
      common/coefficient/
     1  alN2,blN2,clN2,dlN2,elN2,plN2,hlN2,
     2  ahN2,bhN2,chN2,dhN2,ehN2,phN2,hhN2,
     3  alO2,blO2,clO2,dlO2,elO2,plO2,hlO2,
     4  ahO2,bhO2,chO2,dhO2,ehO2,phO2,hhO2,
     5  alAr,plAr,hlAr,
     6  ahAr,phAr,hhAr,
     7  lN2,lO2,lAr
      common/species/
        xmsfN2,xmwN2,xmsfO2,xmwO2,xmsfAr,xmwAr,
     2                 runiv,rmix
c-----------------------------------------------------------------
c  h and cp coefficients from cet85
c
      if(t.le.1000.) then
        hcoefN2=hlN2+alN2*t+blN2*t*t/2.+clN2*t**3./3.+dlN2*
     2          t**4./4.+elN2*t**5./5.
        hcoefO2=hlO2+alO2*t+blO2*t*t/2.+clO2*t**3./3.+dlO2*
     2          t**4./4.+elO2*t**5./5.
        hcoefAr=hlAr+alAr*t
        cpcoefN2=alN2+blN2*t+clN2*t*t+dlN2*t**3.+elN2*t**4.
        cpcoefO2=alO2+blO2*t+clO2*t*t+dlO2*t**3.+elO2*t**4.
        cpcoefAr=alAr
      else
        hcoefN2=hhN2+ahN2*t+bhN2*t*t/2.+chN2*t**3./3.+dhN2*
     2          t**4./4.+ehN2*t**5./5.
        hcoefO2=hhO2+ahO2*t+bhO2*t*t/2.+chO2*t**3./3.+dhO2*
     2          t**4./4.+ehO2*t**5./5.
        hcoefAr=hhAr+ahAr*t
        cpcoefN2=ahN2+bhN2*t+chN2*t*t+dhN2*t**3.+ehN2*t**4.
        cpcoefO2=ahO2+bhO2*t+chO2*t*t+dhO2*t**3.+ehO2*t**4.
        cpcoefAr=ahAr
       end if
c
c  Compute h for each species.  Molar values of h (t=298K)
c     must be found for each species. Naturally occuring
c     species will not have heats of formation.
c
      hN2=hcoefN2*runiv+8669000.
      hO2=hcoefO2*runiv+8682000.
      hAr=hcoefAr*runiv+6196000.45
c
c  Compute h of mixture
```

```
c
      hmix=lN2*hN2*xmsfN2/xmwN2+
     2     lO2*hO2*xmsfO2/xmwO2+
     3     lAr*hAr*xmsfAr/xmwAr
c
c  Compute cp of mixture
c
      cpmix=lN2*cpcoefN2*runiv*xmsfN2/xmwN2
     2     +lO2*cpcoefO2*runiv*xmsfO2/xmwO2
     3     +lAr*cpcoefAr*runiv*xmsfAr/xmwAr
c
c  Compute cv and gamma of the mixture
c
      cvmix=cpmix-rmix
      gmix=cpmix/cvmix

      return
      end
c=============================================================
     subroutine tfromh(h2,tt,t2)
c
c   back calculate temperature given enthalpy using the
c   Newton-Raphson iteration method
c

      common/coefficient/
     1 alN2,blN2,clN2,dlN2,elN2,plN2,hlN2,
     2 ahN2,bhN2,chN2,dhN2,ehN2,phN2,hhN2,
     3 alO2,blO2,clO2,dlO2,elO2,plO2,hlO2,
     4 ahO2,bhO2,chO2,dhO2,ehO2,phO2,hhO2,
     5 alAr,plAr,hlAr,
     6 ahAr,phAr,hhAr,
     7 lN2,lO2,lAr

      common/species/
       xmsfN2,xmwN2,xmsfO2,xmwO2,xmsfAr,xmwAr,
     2              runiv,rmix
c-------------------------------------------------------------
     itmax=300
      tol=1.e-6

      do 100 iter=1,itmax

      if(tt.le.1000.) then

       hcoefN2=hlN2+alN2*tt+blN2*tt*tt/2.+clN2*tt**3./3.
     2       +dlN2*tt**4./4.+elN2*tt**5./5.
       hcoefO2=hlO2+alO2*tt+blO2*tt*tt/2.+clO2*tt**3./3.
     2       +dlO2*tt**4./4.+elO2*tt**5./5.
       hcoefAr=hlAr+alAr*tt
```

```
        else
         hcoefN2=hhN2+ahN2*tt+bhN2*tt*tt/2.+chN2*tt**3./3.
     2          +dhN2*tt**4./4.+ehN2*tt**5./5.
          hcoefO2=hhO2+ahO2*tt+bhO2*tt*tt/2.+chO2*tt**3./3.
     2           +dhO2*tt**4./4.+ehO2*tt**5./5.
          hcoefAr=hhAr+ahAr*tt
         end if

        if(tt.le.1000.) then
          dhcoefN2=alN2+blN2*tt+clN2*tt**2./2.+dlN2*
     2           tt**3./3.+elN2*tt**4./4.
          dhcoefO2=alO2+blO2*tt+clO2*tt**2./2.+dlO2*
     2           tt**3./3.+elO2*tt**4./4.
          dhcoefAr=alAr
        else
          dhcoefN2=ahN2+bhN2*tt+chN2*tt**2./2.+dhN2*
     2           tt**3./3.+ehN2*tt**4./4.
          dhcoefO2=ahO2+bhO2*tt+chO2*tt**2./2.+dhO2*
     2           tt**3./3.+ehO2*tt**4./4.
          dhcoefAr=ahAr
         end if

        hN2=hcoefN2*runiv+8669000.
        hO2=hcoefO2*runiv+8682000.
        hAr=hcoefAr*runiv+6196450.

        hmix=lN2*hN2*xmsfN2/xmwN2+
     2      lO2*hO2*xmsfO2/xmwO2+
     3      lAr*hAr*xmsfAr/xmwAr

        ddhN2=dhcoefN2*runiv
        ddhO2=dhcoefO2*runiv
        ddhAr=dhcoefAr*runiv

        dhmix=-(lN2*ddhN2*xmsfN2/xmwN2+
     2      lO2*ddhO2*xmsfO2/xmwO2+
     3      lAr*ddhAr*xmsfAr/xmwAr)

        fh=h2-hmix
        t2=tt-fh/dhmix
c       write(6,*)'t2 iterated from tfrmoh',t2

        if (abs((tt-t2)/t2).lt.tol) go to 20
        tt=t2
        iter=iter+1
        hmix=0.
  100   continue
        write(6,*)'temperature from enthalpy failed to
     2            converge'
        stop
```

```
   20 continue
      return
      end

c================ end of subroutine enthgam ============
*##############################################################
* WARNING: THIS PROGRAM IS VERY SENSITIVE TO INITIAL
*          GUESSES OF EPSILON.  SHOULD BE SET CLOSE TO
*          ACTUAL VALUE!
*##############################################################
***************************************************************
*   This program will solve the "exact" oblique shock        *
*   problem for an imperfect gas (air), patterned after the*
*   example problem in Zucrow and Hoffman example 7-10.      *
*   The output of this is used to form the IV line that      *
*   is used to validate the imperfect gas model.             *
***************************************************************
      program main

      common/coefficient/
     1   alN2,blN2,clN2,dlN2,elN2,plN2,hlN2,
     2   ahN2,bhN2,chN2,dhN2,ehN2,phN2,hhN2,
     3   alO2,blO2,clO2,dlO2,elO2,plO2,hlO2,
     4   ahO2,bhO2,chO2,dhO2,ehO2,phO2,hhO2,
     5   alAr,plAr,hlAr,
     6   ahAr,phAr,hhAr,
     7   lN2,lO2,lAr
      common/species/xmsfN2,xmwN2,xmsfO2,xmwO2,xmsfAr,xmwAr,
     2                    runiv,rmix
c-------------------------------------------------------------
      tol=1.e-10
      itmax=200

      lN2=1
      lO2=1
      lAr=1
c
c xmw - molecular weight
c xmsf- mass fraction
c a,b,c etc. CET coefficicents (l- low temp, h- high temp)
      xmwN2=28.013
      xmsfN2=.75556737622
      alN2=3.7044177
      blN2=-.14218753e-2
      clN2=.28670392e-5
      dlN2=-.12028885e-8
      elN2=-.13954677e-13
      plN2=2.2336285
      hln2=-.10640795e4

      ahN2=2.8532899
      bhN2=.16022128e-2
      chN2=-.62936893e-6
      dhN2=.11441022e-9
      ehN2=-.78057465e-14
      phN2=6.3964897
      hhN2=-.89008093e3

      xmwO2=32.
      xmsfO2=.231605141
      alO2=3.7837135
      blO2=-.30233634e-2
      clO2=.99492751e-5
      dlO2=-.98189101e-8
      elO2=.33031825e-11
      plO2=3.6416345
      hlO2=-.10638107e4

      ahO2=3.6122139
      bhO2=.74853166e-3
      chO2=-.19820647e-6
      dhO2=.33749008e-10
      ehO2=-.23907374e-14
      phO2=3.6703307
      hhO2=-.11978151e4

      xmwAr=39.944
      xmsfAr=.01282748278
      alAr=2.5
      plAr=4.3660006
      hlAr=-.74537498e3

      ahAr=2.5
      phAr=4.3660006
      hhAr=-.74537502e3

      rmix=287.09
      runiv=8314.34
c
c
      pi=3.14159265359
      degrad=pi/180.

*  initial flow properties

      write(6,*)'input delta in degrees'
      read(5,*) delta
      delta=delta*degrad
      write(6,*)'input temperature in Kelvin'
      read(5,*) t1
      write(6,*)'input mach number'
      read(5,*) xm1
c     write(6,*)'input velocity (m/s)'
c     read(5,*) v1
      write(6,*)'input pressure (N/m^2)'
      read(5,*) p1
c     write(6,*)'input density (kg/m^3)'
c     read(5,*) rho1
      write(6,*)'input first initail guess for epsilon'
      read(5,*) eps1
      eps1=eps1*degrad
      write(6,*)'input second initail guess for epsilon'
      read(5,*) eps2
      eps2=eps2*degrad
*
      rho1=p1/t1/rmix
      call enthgam(t1,hmix,gmix)
      a1=sqrt(gmix*rmix*t1)
      v1=xm1*a1
      h1=hmix
      write(6,*)'delta = ',delta/degrad
      write(6,*)'M1 = ',xm1
      write(6,*)'v1mag = ',v1
      write(6,*)'a1 = ',a1
      write(6,*)'p1 = ',p1
      write(6,*)'t1 = ',t1
      write(6,*)'rho1 = ',rho1
      write(6,*)'gamma1=',gmix
      write(6,*)''

      do 10 iter=1,itmax
      feps1=((gmix+1.)/2.*xm1*xm1/(xm1*xm1*sin(eps1)
     2      *sin(eps1)-1.)-1.)*tan(eps1)
      feps2=((gmix+1.)/2.*xm1*xm1/(xm1*xm1*sin(eps2)
            *sin(eps2)-1.)-1.)*tan(eps2)
      slope=(feps2-feps1)/(eps2-eps1)
      eps3=eps2+(1./tan(delta)-feps2)/slope
c     write(6,*)'epsilon iterations',eps3
      if (abs((eps3-eps2)/eps2).lt.tol) go to 20
      eps1=eps2
      eps2=eps3
      iter=iter+1
      if (iter.gt.itmax) then
      write(6,*)'epsilon did not converge'
      stop
      endif
   10 continue
   20 eps1=eps3
c
   50 xnm1=xm1*sin(eps1)
      xnm2=sqrt((xnm1*xnm1+2./(gmix-1.))/(2.*gmix/(gmix-1.)
     2    *xnm1*xnm1-1.))
      xm2g=xnm2/(sin(eps1-delta))
      rhorat=(gmix+1.)*xnm1*xnm1/(2.+(gmix-1.)*xnm1*xnm1)
      rho21=rho1*rhorat
      trat=(2.*gmix/(gmix+1.)*xnml*xnml-((gmix-1.)/gmix+1.))
      2    *(((gmix-1.)/(gmix+1.))+2./((gmix+1.)*xnml*xnml))
       t2g=trat*t1
       prat=2*gmix/(gmix+1.)*xnml*xnml-((gmix-1.)/(gmix+1.))
       p2g=prat*p1

       vn1=v1*sin(eps1)
       vt=v1*cos(eps1)

       p2=p1+rho1*vn1*vn1*(1.-rho1/rho21)
       h2=h1+vn1*vn1/2.*(1.-(rho1/rho21)*(rho1/rho21))
       call tfromh(h2,t2g,t2)
       rho22=p2/rmix/t2
       if (abs((rho22-rho21)/rho21).lt.tol) go to 40
       p2=p1+rho1*vn1*vn1*(1.-rho1/rho22)
       h2=h1+vn1*vn1/2.*(1.-(rho1/rho22)*(rho1/rho22))
       call tfromh(h2,t2g,t2)

c  Secant method for iteratively finding density

   30 rho23=p2/rmix/t2
       if (abs((rho23-rho22)/rho22).lt.tol) go to 40
       slope=(rho22-rho23)/(rho21-rho22)
       rho2c=rho23+slope*(rho23-rho22)
       if (abs((rho2c-rho23)/rho23).lt.tol) go to 40
       rho21=rho22
       rho22=rho23
       p2=p1+rho1*vn1*vn1*(1.-rho1/rho2c)
c      write(6,*)'iterations of p',p2
       h2=h1+vn1*vn1/2.*(1.-(rho1/rho2c)*(rho1/rho2c))
       call tfromh(h2,t2g,t2)
       go to 30

   40 continue
       rho2=rho2c
       vn2=rho1/rho2*vn1
       v2=sqrt(vt*vt+vn2*vn2)
       call enthgam(t2,hmix,gmix)
       a2=sqrt(gmix*rmix*t2)
       xm2=v2/a2
c      write(6,*)'mach2=',xm2
       eps2=delta+asin(vn2/v2)
       if (abs((eps2-eps1)/eps1).lt.tol) go to 60
       eps1=eps2
       go to 50
   60 continue

       write(6,*)'epsilon =',eps2/degrad
       write(6,*)'M2 =',xm2
       write(6,*)'v2mag =',v2
      write(6,*)'a2 =',a2
      write(6,*)'p2 =',p2
      write(6,*)'t2 =',t2
      write(6,*)'rho2 =',rho2
      write(6,*)'gamma2 =',gmix
      stop
      end
c
```

Bibliography

1. Anderson, John D. Hypersonics and High Temperature Gas Dynamics. New York: Mcgraw-Hill, Inc., 1989.

2. Bussing, Thomas R. A. and Scott Eberhardt. "Chemistry Associated with Hypersonic Vehicles," AIAA, Paper 87-1292: 1-9 (July 1989).

3. Doty, John H. Maximum Thrust Planar Supersonic Nozzles Using a Flux-Difference-Splitting Technique. PhD Dissertation. Purdue University, Lafayette, Indiana, 1991.

4. Gordon, Sanford and Bonnie J. McBride. Computer Program for Calculation of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks, and Chapman-Houquet Detonations. NASA SP- 273. NASA Lewis Research Center, Interim Reivision, March 1976, CET89 Update, 1989.

5. John, James E. A. Gas Dynamics. Boston: Allyn and Bacon, 1969.

6. Pandolfini, P. Instructions for Using RAMJET Performance Analysis (RJPA)- IBM-PC Version 1.0. Contract JHU/APL-NASP-86-2. Laurel MD: The John Hopkins University Applied Physics Laboratory, November, 1986.

7. Snelling, Capt Sandra L. Effect of Non-uniform Entrance Flow Profile on Hypersonic Nozzle Pitching Moment. MS Thesis, AFIT/GA/ENY/91D . School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1991.

8. Tannehill, John C. and John O. Ievalts. "An Upwind Parabolized Navier-Stokes Code for Chemically Reacting Flows," AIAA Paper 88-2614: 1-14 (June, 1988).

9. Vincenti, Walter G. and Charles H. Kruger, Jr. Introduction to Physical Gas Dynamics. Malabar, Florida: Robert E. Krieger Publishing Company, 1986.

10. Zucrow, Maurice J. and Joe D. Hoffman. Gas Dynamics, Volume I. New York: John Wiley and Sons, Inc., 1976.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1991	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
IMPROVEMENT OF THE THERMODYNAMIC MODEL FOR A FLUX-DIFFERENCE-SPLITTING (FDS) ALGORITHM FOR COMPUTATION OF HIGH SPEED FLOWS

5. FUNDING NUMBERS

6. AUTHOR(S)
Schieve, Mark E., Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GAE/ENY/91D-10

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The thermodynamic model of a previously existing first-order accurate Flux-Difference-Splitting (FDS) algorithm for planar, supersonic nozzles is modified. The thermodynamic model is changed from a calorically and thermally perfect gas to a thermally perfect (imperfect) gas, where the flow field is "frozen" or non-reacting. Using curve fittings of JANAF thermochemical data, the code can handle nine gas species, as well, to model combustion products entering the nozzle inlet. The marching scheme is not altered.

An oblique shock reflection study is done to validate the improved gas model. A low temperature case and a high temperature case are run. For the first case, the perfect and imperfect models are nearly identical. For the more extreme case, pressure for the perfect gas is 9.4% greater than the exact solution, at the upper boundary, across the shock. An interior flow nozzle is run for the two cases, with air as the working fluid. Again, the two models give identical results for the low temperature case. For the high temperature case, integrated nozzle thrust for the imperfect gas model is 16% higher than that for the original perfect gas model.

14. SUBJECT TERMS
Hypersonic Nozzle, Riemann Solution, NASP, SCRAMJET, Frozen Flow, Thermodynamic Model, Propulsion, Flux-Difference-Splitting

15. NUMBER OF PAGES
108

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL